Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Servicio web REST de gestión de eventos con Java Spring Framework - Panel de gestión web

Autor: Adrián Gil Gago
Tutor: Teresa Ariza Gómez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías de Telecomunicación

Servicio web REST de gestión de eventos con Java Spring Framework - Panel de gestión web

Autor:
Adrián Gil Gago

Tutor:
Teresa Ariza Gómez
Profesor titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Servicio web REST de gestión de eventos con Java Spring Framework - Panel de gestión web

Autor: Adrián Gil Gago

Tutor: Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

En primer lugar, quiero agradecer a mi pareja, que me ha ayudado con todo lo que ha podido y más, para que a día de hoy pueda presentar este trabajo.

En segundo lugar, agradecer a mi familia y amigos de siempre, todo el apoyo que también me han aportado, así como todas las experiencias vividas que hemos tenido juntos y que me han llevado a ser como soy y a poder estar donde estoy.

Finalmente, quiero agradecer a todos los profesores que he tenido en esta facultad, aunque a algunos los haya conocido de manera fugaz, todos los conocimientos y valores que me han transmitido en mi paso por la universidad, ya que espero y estoy convencido de que los mismos me serán de gran ayuda en mi futuro profesional y personal.

Adrián Gil Gago

Alumno de la Escuela Técnica Superior de Ingeniería

Sevilla, 2016

Resumen

Este proyecto es una parte de un todo que se encuentra dividido en dos. Por una parte tenemos una aplicación desarrollada en Android para dispositivos móviles y su correspondiente servicio web REST, en la que se pueden visualizar y filtrar eventos de cualquier índole, que a su vez se celebran en lugares denominados locales.

Por la otra, se encuentra la parte de la que nos ocupamos nosotros, que es la interfaz gráfica de gestión de eventos o panel de gestión web, de la que hace uso el propietario de los locales y de los eventos para gestionar los mismos, así como toda su información de usuario.

El sistema completo de panel de gestión web consta de una base de datos relacional MySQL, donde se almacenarán los datos de usuario, así como de un servicio web REST desarrollado en Java Spring Framework, que coordinará las operaciones sobre la base de datos, y para finalizar una interfaz gráfica de usuario desarrollada en HTML5, en conjunto con CSS y JavaScript, además de otras extensiones y librerías adicionales de éste último.

Índice

ÍNDICE DE FIGURAS

1 ASPECTOS GENERALES

Los que no puedan mantener el ritmo de la revolución tecnológica, se encontrarán con que ellos mismos se han vuelto obsoletos

- Katherine Neville -

1.1 Motivación y objetivos

Este es uno de un conjunto de dos proyectos que pretenden realizar una aproximación lo más fiel posible a un sistema real y funcional de gestión y promoción de eventos. El proyecto que nos ocupa pretende abordar el diseño y desarrollo de la parte del sistema dedicada a la gestión de eventos por parte de los propietarios de los mismos. Toda la aplicación web se desplegará desde un sólo fichero gracias a la ayuda del framework que hemos escogido, Spring, que detallaremos más adelante. Además esto hace que su despliegue sea sencillo tanto en máquinas locales como en servicios en la nube.

Si bien es cierto que toda la aplicación se ha realizado dentro de los estándares que se espera que cumpla un proyecto de esta universidad, hay ciertos aspectos a los que hemos dedicado especial atención, debido principalmente a su componente de diferenciación con respecto a otros proyectos que se hayan realizado con anterioridad.

Tal y como se deduce de lo que acabamos de exponer previamente, este aspecto al que hemos dedicado especial atención es al servicio web REST desarrollado completamente en Spring Framework, usando como lenguaje de programación Java. Durante el desarrollo del mismo no se ha utilizado ni una sola biblioteca nativa de Java, salvo en el manejo de tipos simples, con el fin de aprovechar y mostrar al máximo las capacidades de Spring.

Además de esto, el crecimiento exponencial de Spring en el mercado, la alta versatilidad que nos ofrece y su estrecha relación con los SOA (*arquitecturas orientadas a servicios*, en español), lo hacen sumamente interesante de cara a su estudio en un proyecto de Grado en Ingeniería de Tecnologías de Telecomunicación.

Finalmente, en cuanto al sistema de gestión de eventos como aplicación funcional, es nuestro deseo, de ambos alumnos con sendos proyectos, el poder seguir desarrollándola fuera de los límites de la Universidad, ya que el contexto es propicio al no haber una opción clara y bien definida en el mercado, aunque ya sabemos a priori, que esta será una ardua tarea y de una dificultad extraordinaria. A continuación explicaremos el contexto y demás aspectos que encontramos relacionados con esta cuestión.

Una vez presentadas y detalladas las motivaciones que nos han llevado a acometer la realización de este proyecto, pasaremos a explicar en este capítulo tanto el contexto, como el problema y la solución que nosotros proponemos, en materia de gestión de eventos, así como ventajas y desventajas de los antecedentes relacionados con la ejecución de esta tarea.

1.2 Contexto y presentación del problema

El auge de los dispositivos móviles portátiles, hace que llevemos al lugar donde nos desplacemos la herramienta más potente de trabajo y de ocio que existe actualmente: Internet. Y es sabido, que el punto principal de dichos dispositivos, no es ni más ni menos que la socialización. No estaba equivocado Aristóteles hace cientos de años, cuando definió al ser humano como un animal social, Zoon politikón.

Por esto tenemos que a día de hoy, las aplicaciones que disfrutan de mayor popularidad son aquellas con cometido social, bien sean WhatsApp, Facebook, Instagram, Snapchat y un largo etcétera, pasando también por aquellas que abordan no sólo los temas relacionados con la socialización, como Tinder, Badoo o Grindr.

Debido a lo que acabamos de explicar, hay un hueco de mercado en el que nosotros nos centraremos, relacionado con lo social: los eventos. Es sumamente llamativo que a día de hoy no exista una plataforma predominante cuya finalidad sea única y exclusivamente los eventos, que los ofrezca de manera sencilla, rápida y no a través de otros servicios o como un plugin o adición a un servicio cuyo propósito es otro.

Esto además, viene acrecentado porque en las grandes urbes sea el coloquialmente denominado “boca a boca” la única forma de transmisión de que un concierto, una muestra especial en un museo o una clase de cocina van a celebrarse en dicho lugar. Si le añadimos el hecho de que las personas que viajan a un lugar y no lo han visitado nunca, no conocen a nadie, pues este problema aumenta aún más.

Finalmente, añadiendo a esta carencia nuestro frenético ritmo de vida actual, principalmente relacionado con las condiciones de trabajo, hace que tengamos que aprovechar nuestro tiempo libre haciendo aquello que más nos guste y utilizando el menor tiempo posible en encontrarlo.

Por todo ello, una plataforma de gestión de eventos puede convertirse en fundamental como siguiente paso en la mejora de la calidad de vida de las personas en general.

1.3 Antecedentes

Dentro de este apartado pasaremos a presentar los servicios y aplicaciones en los que nos hemos inspirado para poder elaborar nuestro proyecto.

1.3.1 Tuenti

Tuenti fue una red social cuya popularización fue muy alta, llegando en ocasiones a cifras de crecimiento del 200% anual en el período 2006/2009, siendo la red social más popular en España durante ese período. Su compra por parte de la multinacional española Telefónica y una posterior mala gestión la hicieron entrar en una serie de períodos de transición en el que no acabó de encontrar su función real, hasta que la expansión de Facebook en España la barrió del mercado. Actualmente es una operadora móvil virtual (OMV).

Figura 1-1 Fragmento de la interfaz web de Tuenti en la que nos hemos inspirado.

Lo que hemos tomado de Tuenti ha sido la idea de gestionar los eventos como plataforma y la simplicidad de su diseño cuando comenzó a popularizarse, ya que en nuestra opinión ha sido el mejor de cuantos ha tenido.

También el hecho de cómo se gestionaban los formularios de modificación de datos de usuario y otros aspectos relacionados con el mismo.

1.4 Descripción de la solución

En este apartado presentaremos las principales características y funcionalidades de la solución que hemos decidido proponer nosotros.

1.4.1 Objetivos específicos y funciones

El objetivo principal de nuestra solución es facilitar a los usuarios el registro y la gestión de eventos de la manera más sencilla posible, sin personal externo que le asista en la misma a través de un gestor con una interfaz gráfica web. Por lo tanto, el usuario tendrá disponible en cada página de la ya citada web una serie de breves explicaciones que, sin molestar a su visualización, le ofrecerán información adicional correspondiente a cada página del sitio.

El usuario podrá dar de alta su perfil a través de la interfaz de registro disponible, especificando sus datos de usuario y para que así le sea posible con posterioridad acceder a las funcionalidades de la aplicación.

Una vez dado de alta el usuario, podrá gestionar sus eventos y sus locales, en una interfaz en la que predominan los formularios. Es necesario resaltar en este punto que para el manejo de formularios o su comprobación no se ha usado ni una sola línea de PHP o cualquier lenguaje ajeno a los ya descritos previamente.

Dentro de la gestión de locales, un usuario puede añadir locales, especificando su localización a través de la API de Google Maps, así como otra información relacionada con el mismo. También puede añadir eventos a cada local, que a su vez heredarán la localización en el caso del local, ya que un local es aquel sitio donde se celebra el evento. A cada evento, también se le podrán añadir sus datos correspondientes.

Respecto a la gestión de la seguridad, hemos optado por utilizar unas cookies de sesión en conjunto con otras de protección frente a ataques CSRF y si bien es cierto que nuestra aplicación a día de hoy puede presentar vulnerabilidades, hemos optado por aquello que hemos podido implementar, debido al hecho de que todo se ha realizado utilizando librerías nativas de Spring Framework.

Cabe destacar que uno de los puntos fuertes de este trabajo no son las funcionalidades que se ofrecen al usuario en sí, sino cómo se ofrecen mediante Spring Framework. En el apartado de tecnologías se observará esta parte más detenidamente, con lo que el lector de este trabajo puede encontrarse que la gestión de eventos y Spring se intercambian los roles de fin y medio como quizá no se suponía en un principio.

1.4.2 Arquitectura de nuestra aplicación

La arquitectura que hemos implementado en nuestra solución presenta los siguientes elementos, que se describen en la Figura 1-2.

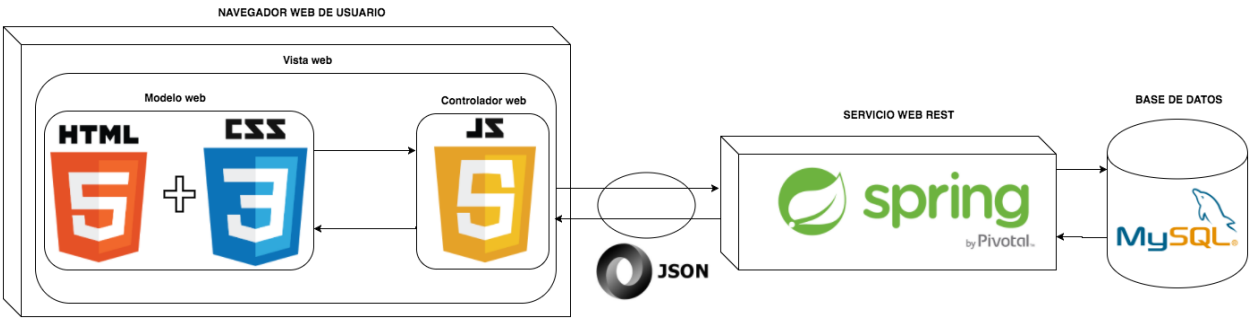


Figura 1-2 Arquitectura general de nuestra aplicación.

- **Navegador web de usuario**: obtiene los datos del servidor y en él muestra la vista, que se genera modificando el modelo web, que consta a su vez de los ficheros HTML5 y CSS3, con los controladores, que son ficheros JavaScript que obtienen del servicio web REST los datos a representar en dicho modelo y los imprimen o envía.
- **Servicio web**: Servicio web con arquitectura REST, desarrollado enteramente mediante el uso del framework Java Spring Framework. Su función principal es coordinar la autenticación de usuarios y la realización de operaciones CRUD (*Create, Read, Update and Delete*, en inglés) de usuarios, locales y eventos. Cabe destacar, que el formato de ficheros de intercambio entre el controlador web JavaScript y el servicio web Spring por el que nosotros hemos optado ha sido JSON.
- **Base de datos**: La base de datos es el lugar donde almacenaremos todos los datos relacionados con información de usuarios, locales y eventos. Está diseñada siguiendo un modelo entidad-relación y usando el gestor de Bases de Datos MySQL.

1.5 Bibliografía y referencias

La bibliografía y referencias que hemos consultado para la realización de este trabajo son exclusivamente recursos electrónicos, es decir, todos se encuentran en Internet y están en continua actualización. Tenemos dos motivos para escogerlos en lugar del formato tradicional en papel: el primero es que Spring dispone de una magnífica documentación en línea y no sólo eso, sino que también ofrece tutoriales paso a paso y en otro formato que ellos denominan guías, con lo que no es necesario acceder al formato papel para obtener información del framework. El segundo motivo es que Spring se encuentra en evolución constante por la gran aceptación que está teniendo entre la comunidad de desarrolladores en general, por lo que cualquier libro de Spring que lleve publicado más de seis meses es posible que haga uso de librerías o funciones que se encuentren a día de hoy caducadas (*deprecated*, en inglés), bien porque las hayan sustituido por una versión nueva y mejor o bien porque se hayan dejado de utilizar por cuestiones de seguridad. Las referencias web que hemos consultado para la elaboración son las siguientes:

En relación a MySQL y módulos relacionados:

- Web oficial de MySQL: https://www.mysql.com/

En relación a Spring Framework y módulos relacionados:

- Documentación de Spring: https://spring.io/docs
- Guía de Spring: https://spring.io/guides
- Web de tutoriales de propósito general: http://www.mkyong.com/tutorials/
- Web de tutoriales especializada en Spring: http://www.baeldung.com/

En relación a HTML5:

- Referencia de HTML5 del W3C: https://dev.w3.org/html5/html-author/
- Guía de HTML5 de MDN: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
- Tutoriales de HTML5 de W3Schools: http://www.w3schools.com/html/

En relación a CSS3:

- Tutoriales de CSS3 de W3Schools: http://www.w3schools.com/css/css3_intro.asp

En relación a JavaScript y librerías relacionadas:

- Tutoriales de JavaScript de W3Schools: http://www.w3schools.com/js/
- Web oficial de jQuery: https://jquery.com/
- Web oficial de jQuery Validation Plugin: https://jqueryvalidation.org/
- Perfil de Github de Jörn Zaefferer (librerías de traducción de jQVP): https://github.com/jzaefferer/
- Perfil de Github de Klaus Hartl (librerías de cookies): https://github.com/carhartl/

1.6 Estructura de la memoria

A continuación, pasaremos a detallar los capítulos de los que consta la memoria, cada uno con un breve resumen para que el lector pueda conocer de un vistazo de qué trata cada uno:

1. **Aspectos generales**: Consta de las razones por las que hemos llevado a cabo el proyecto, así como una aproximación genérica a los detalles del mismo.
2. **Tecnologías utilizadas**: Consta de un pequeño resumen de cada tecnología que hemos utilizado en el desarrollo de nuestra aplicación, así como el motivo por el que hemos hecho uso de cada una de ellas.
3. **Herramientas de desarrollo**: Consta de un pequeño resumen de cada herramienta que hemos utilizado en el desarrollo de nuestra aplicación, así como el motivo por el que hemos hecho uso de cada una de ellas.
4. **Arquitectura y análisis**: Consta de un examen de la funcionalidad de la aplicación desde un punto de vista de observación descriptivo a través de una serie de diagramas de diversos tipos.
5. **Funcionalidad de la aplicación**: Consta de un examen de la funcionalidad de la aplicación a través de lo que puede observar un usuario de la misma.
6. **Líneas de mejora y conclusiones**: Presentamos una serie de tecnologías que podrían utilizarse para mejorar nuestra aplicación, así como de las razones de cada una. Al final del capítulo se incluyen las conclusiones sacadas en claro después de la realización del proyecto.
7. **Introducción a los anexos**: Es una breve reflexión que incluyen los motivos de la inclusión de los Anexos como tales en la memoria.

Después de los capítulos tenemos disponibles los Anexos, en los que se detalla información adicional del proyecto:

- **Anexo A**: Se detalla cómo realizar el despliegue de un servicio desarrollado con Spring Boot en un servicio en la nube; Heroku.
- **Anexo B**: Fichero en SQL de creación de la estructura de la base de datos que hemos utilizado en nuestro proyecto.
- **Anexo C**: Código fuente en Java del servicio web REST desarrollado en Spring Framework.
- **Anexo D**: Ficheros de configuración en XML y texto plano del servicio web REST desarrollado en Spring Framework.
- **Anexo E**: Ficheros de código fuente en HTML5 utilizados en la programación de la interfaz gráfica web.
- **Anexo F**: Ficheros de código fuente en CSS3 utilizados en la programación de la interfaz gráfica web.
- **Anexo G**: Ficheros de código fuente de las librerías de JavaScript utilizadas en la programación de la interfaz gráfica web. No se incluyen las librerías de jQuery, jQuery Validator ni las de traducción de mensajes de aviso de este último, ya que constan de tantas líneas que no son legibles. Por otra parte, sí que se incluye la librería de manejo de cookies de Klaus Harl, de las que hemos hecho uso, ya que esta sí que es legible, a pesar de que podamos encontrarla en Internet.
- **Anexo H**: Ficheros de código fuente de los controladores en JavaScript utilizados en la programación de la interfaz gráfica web.

2 TECNOLOGÍAS UTILIZADAS

El gran motor del cambio, la tecnología.

- Alvin Toffler -

Tal y como reza la frase de nuestro encabezado, la base de la revolución humana es la tecnología, de hecho es la técnica a fin de cuentas lo único que nos diferencia de otros seres vivos. Es un aspecto fundamental en el día a día del ser humano y no lo podía ser menos en nuestro trabajo, debido a su carácter tecnológico innovador. En este capítulo nos centraremos en todas las tecnologías de las que hemos hecho uso para llevar a cabo la realización de nuestra aplicación. Explicaremos en qué consisten y asimismo, en qué aspectos de nuestro proyecto las hemos utilizado y con qué fin, entrando más en detalle en un capítulo posterior, en el que pormenorizaremos de qué manera se han implementado en nuestra aplicación.

2.1 Intercambio de datos - JSON

2.1.1 Introducción

JSON (en inglés, *JavaScript Object Notation*) es un tipo de formato para la representación de datos siguiendo una sintaxis específica, cuya función fundamental consiste en que diferentes aplicaciones puedan intercambiar información independientemente de la plataforma o lenguaje en el que han sido desarrolladas.

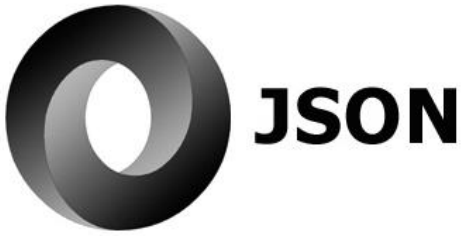


Figura 2-1 Logo gráfico de JSON.

2.1.2 Uso en nuestro proyecto

Tal y como puede observarse en la Figura 1-2, hemos optado por JSON como estándar de transmisión de datos en cliente y servicio web. Esto es debido a tres factores importantes, que presentamos a continuación:

- Su análisis sintáctico (*parsing*, en inglés) y su generación son relativamente sencillas en JavaScript, lenguaje de programación en el lado del cliente que nosotros utilizamos.
- Si bien es cierto que nuestra arquitectura de servicio web, que es REST, soporta también otros lenguajes de representación de datos como XML, Spring Framework en su última versión utiliza JSON como formato estándar de intercambio de datos en desarrollo de servicios web REST.
- Luego teniendo estos dos motivos, el hecho de utilizarlo lo hacen completamente recomendable y nos ahorra muchas complicaciones en cuanto a transmisión de información cliente-servidor.

2.2 Base de datos

2.2.1 MySQL

2.2.1.1 Introducción

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como el sistema de base datos Open Source más popular del mundo.

Figura 2-2 Logo gráfico de MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Por lo tanto, MySQL está licenciado bajo un sistema de doble licenciamiento.

2.2.1.2 Uso en nuestro proyecto

Tal y como puede observarse en Figura 2-2, hemos optado por MySQL como opción de almacenamiento de todos los datos de la aplicación: esto significa que utilizaremos dicho gestor con el fin de que guarde toda la información relacionada con usuarios, locales y eventos. El hecho de que hayamos elegido esta tecnología se debe a diversos factores:

- La disponibilidad de una interfaz de conexión a base de datos MySQL compatible con Spring, denominada JDBC.
- La alta popularidad de la que goza en Internet y el mundo de la computación en general.
- La capacidad de gestionarla y diseñarla fácilmente mediante las herramientas de asistencia XAMPP y MySQLWorkbench respectivamente.

2.3 Servicio web REST

2.3.1 Arquitectura REST

2.3.1.1 Introducción

La Transferencia de Estado Representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

Para que un servicio sea REST debe cumplir una serie de requisitos.....

Hay un punto importante a considerar en cuanto a servicios web REST. En la práctica se distingue entre lo que se denomina REST y lo que se denomina RESTFUL, aunque haya algunos desarrolladores que usen ambos términos indistintamente. El origen de esta teoría es debido a una teoría no aprobada por Roy Fielding, el que podríamos considerar creador de REST, sino más bien por una serie de prácticas que han establecido un estándar de facto respecto a esta distinción. Este estándar se puede observar en la pirámide de abajo, que nos muestra cómo hay servicios REST por niveles aunque no cumplan todas las especficaciones de Roy Fielding y, en el caso de que un servicio web cumpla todos los niveles, podríamos calificar dicho servicio web REST como RESTFUL.

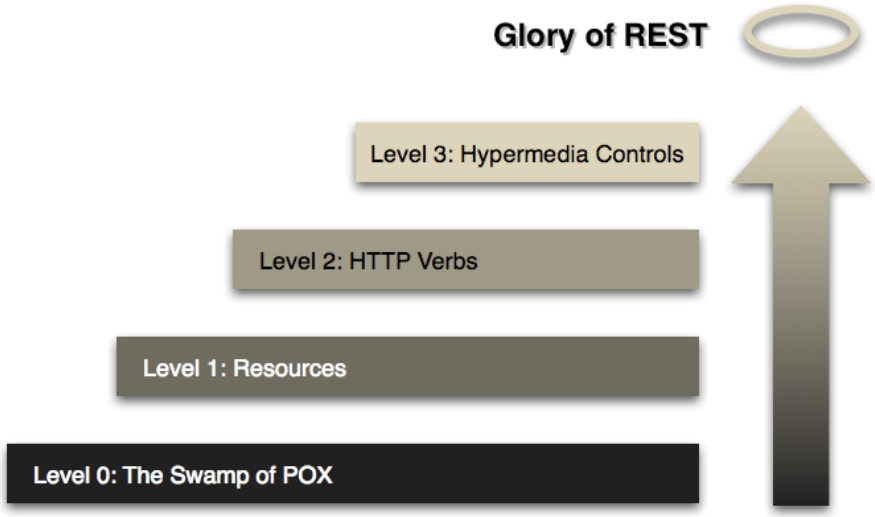


Figura 2-3 Pirámide de niveles REST.

2.3.1.2 Uso en nuestro proyecto

Tal y como se observa en la Figura 1-2, hemos utilizado REST como modelo de diseño para el servicio web. Esto no quiere decir que cumplamos todos los principios, ya que como se ha comentado justo en el punto anterior, un servicio web se considera REST aunque no cumpla todos los principios descritos. El motivo principal por el que hemos escogido esta arquitectura para realizar el diseño de nuestro servicio web es por la alta popularidad de la que goza en Internet a día de hoy en la industria y entre los desarrolladores, ya que si nos ceñimos a diferencias de rendimiento, como por ejemplo frente a su principal competencia, que es SOAP, si bien es cierto que REST mejora el rendimiento respecto a

SOAP, su mejora no suele ser muy significativa.

2.4 Spring Framework

2.4.1 Spring

2.4.1.1 Introducción

Spring es un framework que se utiliza para el desarrollo de aplicaciones y a su vez es un contenedor de inversión de control, es de código abierto y utiliza Java como lenguaje de programación.

Figura 2-4 Logo gráfico de Spring.

La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro Expert One-on-One J2EE Design and Development (Wrox Press, octubre 2002). El framework fue lanzado inicialmente bajo la licencia Apache 2.0 en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. La versión 1.2.6 de Spring Framework obtuvo reconocimientos Jolt Awards y Jax Innovation Awards en 2006. Spring Framework 2.0 fue lanzado en 2006, la versión 2.5 en noviembre de 2007, Spring 3.0 en diciembre de 2009, y Spring 3.1 dos años más tarde. El inicio del desarrollo de la versión 4.0 fue anunciado en enero de 2013.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para facilitar la construcción de aplicaciones web complejas sobre Java. A pesar de que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

2.4.1.2 Uso en nuestro proyecto

Tal y como se observa en la Figura 1-2, hemos utilizado Spring como framework de construcción de nuestro servicio web REST, ya que aporta una gran cantidad de herramientas que elimina ciertas complejidades asociadas con el desarrollo de servicios web.

De entre todos los módulos de Spring Framework (figura 3-5), hemos utilizado Spring Boot, Spring Security, Spring Relational Data Access (mediante el patrón DAO) y Spring Web.

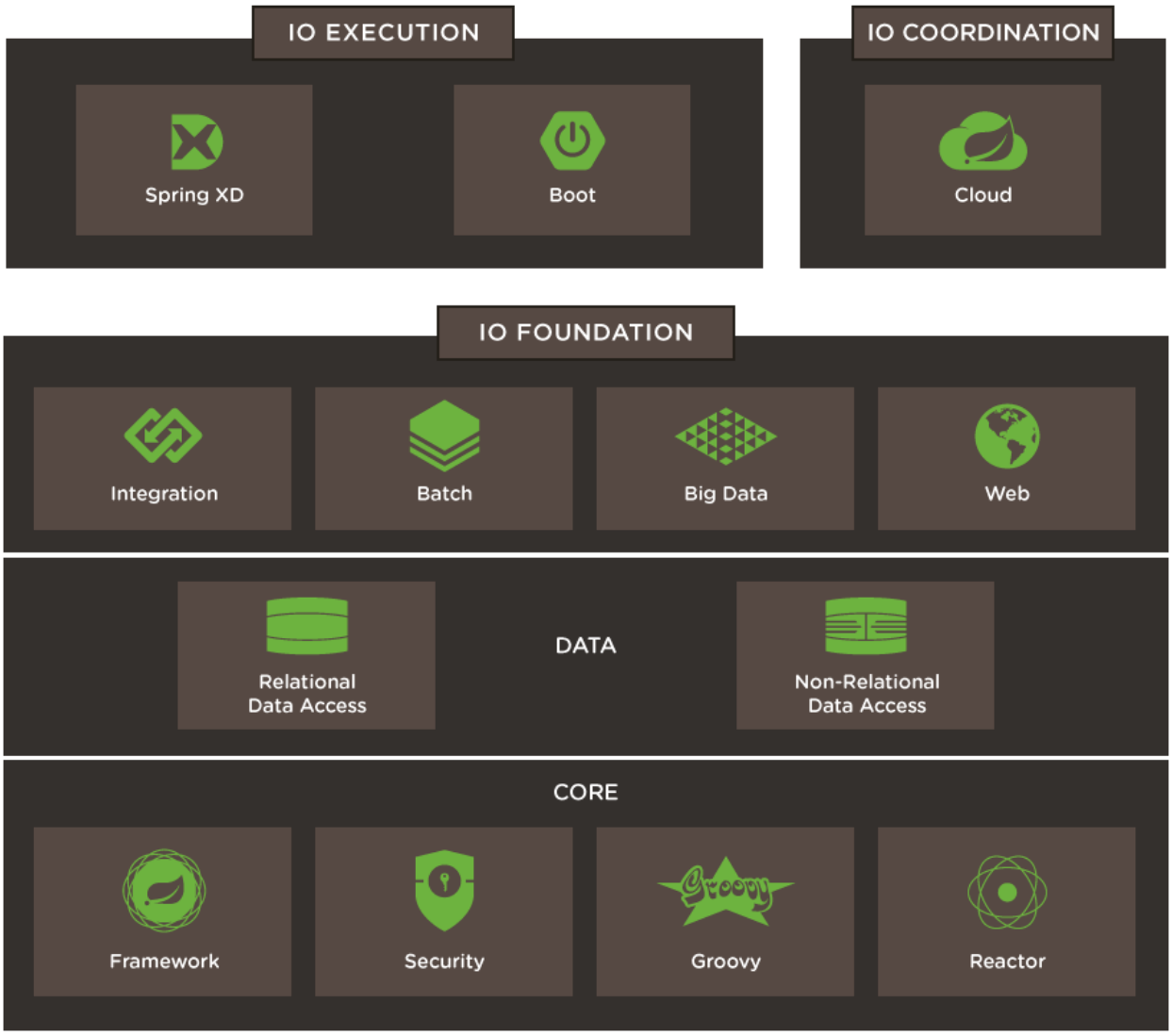


Figura 2-5 Representación gráfica de los módulos de Spring.

2.4.2 Maven

2.4.2.1 Introducción

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation.

Figura 2-6 Logo gráfico de Maven

Maven utiliza un fichero denominado Project Object Model (POM) para describir el proyecto software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.sonatype.mavenbook.simple</groupId>
    <artifactId>simple</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>simple</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Figura 2-7 Estructura típica de un fichero POM.xml.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven, pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción

de la aplicación, para que puedan acceder todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

2.4.2.2 Uso en nuestro proyecto

Aunque no se observe en la Figura 1-2, hemos utilizado Maven para producir los ficheros JAR del servicio web. Hemos optado por esta herramienta para la construcción de nuestra aplicación porque al definir en el fichero POM.xml de configuración de Maven propio de nuestro proyecto las dependencias de librerías que necesitamos, procede a su descarga y compilación automáticamente al generar el JAR. Además, su buena integración con el IDE personalizado de Spring (Spring Tools Suite, del que hablaremos más tarde), hace que vaya comprobando o nos sugiera en tiempo real aquellas librerías de las que hacemos o deseemos hacer uso.

2.4.3 Spring Boot

2.4.3.1 Introducción

Spring Boot es un módulo de Spring que nos permite de manera sencilla, crear aplicaciones autónomas de estilo profesional basadas en Spring, que simplemente hay que "correr" (*just run*, en su documentación en inglés). Básicamente consiste en un mecanismo centrado en una clase principal, que tiene una configuración por defecto y que podemos personalizar. Es compatible con Maven y Gradle y su formato por defecto para el despliegue es JAR, aunque también soporta WAR.

Figura 2-8 Logo gráfico de Spring Boot.

2.4.3.2 Dónde lo hemos utilizado en nuestro proyecto

Aunque no se observe en la Figura 1-2, hemos utilizado Spring Boot para la creación de nuestra aplicación web. Hemos configurado una clase principal que se encarga de configurar la mayor parte de parámetros de despliegue por nosotros, de tal forma que en un sólo fichero JAR tenemos toda la aplicación completa (servicio web REST y código web HTML5, CSS3 y JavaScript) y con tan sólo ejecutar el fichero JAR como si fuera un fichero normal de clases, se produce el lanzamiento y despliegue de la aplicación de manera automática, incluyendo un servidor web Tomcat embebido.

2.4.4 Spring Security

2.4.4.1 Introducción

Spring Security es un módulo de Spring que nos permite, securizar aplicaciones Spring desde el punto de vista de la autenticación o lo que es lo mismo, comprobar que el usuario exista y permitir el acceso a sus recursos, y la autorización, es decir, distinguir entre tipos de usuarios según sus credenciales. Como en el caso anterior, consta de una serie de clases y parámetros que hay que personalizar y adecuar para que se ajuste de la mejor manera posible a nuestro proyecto.

Figura 2-9 Logo gráfico de Spring Security

2.4.4.2 Uso en nuestro proyecto

Aunque no se observe en la Figura 1-2, hemos utilizado Spring Security para la autenticación y autorización de acceso a nuestra aplicación web a través de Internet.

2.4.5 Inversión de Control (IoC)

2.4.5.1 Introducción

Se refiere a todo aquel diseño de software cuyo propósito obedece a la necesidad de querer controlar el flujo de ejecución de este, de forma automática y transparente, es decir, ceder el control de ese flujo a un "agente externo", normalmente un framework.

Spring Framework, ya implementa un "Contenedor IoC", que se encarga de gestionar las instancias (así como sus creaciones y destrucciones) de los objetos del usuario. Por tanto las aplicaciones que utilicen el framework de Spring (no Spring propiamente dicho) utilizarán Inversión de Control.

2.4.5.2 Dónde lo hemos utilizado en nuestro proyecto

Tal y como se ha comentado con anterioridad, este concepto software lo utilizamos de manera transparente mediante el uso de Spring Framework en el desarrollo de nuestro servicio web REST.

2.4.6 Inyección de Dependencias (DI)

2.4.6.1 Introducción

DI es un patrón de diseño que sirve para "inyectar" componentes a las clases que tenemos implementadas. Esos componentes son contratos que necesitan nuestras clases para poder funcionar, de ahí el concepto de "dependencia". La diferencia sustancial en este patrón de diseño es que nuestras clases no crearán esos objetos que necesitan, sino que se les suministrará otra clase "contenedora" perteneciente al Framework DI que estemos utilizando y que inyectará la implementación deseada a nuestro contrato, y todo ello sin tener que hacer un solo "new".

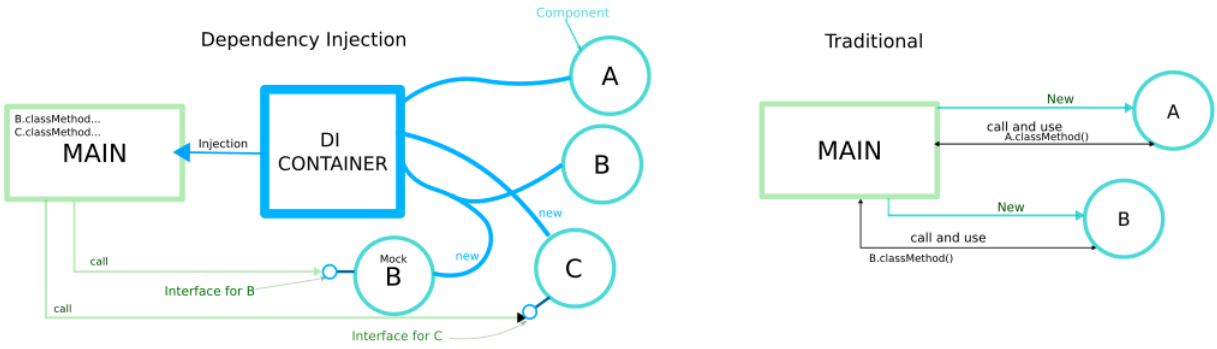


Figura 2-10 Forma tradicional de programación comparada con DI.

Pese a que la inyección de dependencias puede realizarse referenciando directamente las clases de dichas dependencias, esa no es una buena práctica dado que sus componentes tienen una fuerte relación entre sí, que al final nos supondrá un inconveniente para poder mantener nuestro software. Para resolver ese problema normalmente los frameworks DI utilizan en la inyección de dependencias Interfaces entre componentes.

Al componer esa relación entre interfaces conseguimos abstraer la relación entre una clase A que depende de una clase B sin importar la implementación de cada uno de los dos. A ese concepto le llamamos desacoplamiento y está presente en el mandato DIP (Dependency Injection Principle) de los cinco que componen los principios SOLID.

El poco acoplamiento nos permite tener la relación entre nuestras clases separadas por capas de abstracción (Interfaces). Por lo tanto, al no tener una relación directa entre los diferentes módulos de nuestro software evitamos que un cambio interno en uno de ellos afecte a toda la aplicación, de ese modo podemos arreglar el problema directamente cambiando las líneas de implementación conflictivas de la clase en cuestión.

Finalmente, hay que tener en cuenta la reutilización modular. Con la inyección de dependencias podemos reutilizar nuestros componentes en toda la aplicación debido a su poco acoplamiento. Bajo esta premisa nos obligamos a programar nuestras clases bajo el llamado "Single Responsability Principle" concretando así el principio de única responsabilidad en el comportamiento de una clase, haciendo de esta un componente más reutilizable.

2.4.6.2 Uso nuestro proyecto

Nosotros hemos utilizado Inyección de Dependencias en nuestro proyecto en conjunción con IoC mediante la definición de Beans de Spring. Estos Beans que hemos definido son los relacionados con el DataSource de conexión a la base de datos y los DAO de propietarios, locales y eventos, con el fin de aprovechar al máximo lo que esta capacidad de Spring nos ofrece.

2.4.7 Spring Beans

2.4.7.1 Introducción

A diferencia de los Beans convencionales que representan una clase con un cierto formato, la particularidad de los Beans de Spring es que son objetos creados y manejados por el contenedor Spring. El contenedor se encuentra en el núcleo del marco de trabajo de Spring y utiliza inyección de dependencias para gestionar los componentes que forman la aplicación. Se encarga de varias tareas, como crear, conectar y alojar los objetos definidos por los Beans. Además hace de dispensador proporcionando Beans por petición. El contenedor carga las definiciones de Beans escritas en archivos XML estructurados de forma ordenada. Tipos de contenedor de Spring:

- Fábrica de Beans (Bean Factory): contenedor sencillo con soporte básico de inyección de dependencias.
- Contexto de aplicación (Application Context): es una implementación de la Bean Factory que proporciona opciones avanzadas como por ejemplo: medios para resolver mensajes de texto, publicación de Beans registrados como receptores o formas genéricas de abrir recursos de archivo.

Otra diferencia de los Beans de Spring es que a éstos se añade un ciclo nuevo para que el Bean sepa cuál es su contexto de aplicación. Podemos ordenar las fases de la vida de un Bean de la siguiente forma:

- Instanciación
- Inyección de las propiedades
- Nombre del Bean
- Nombre de la fábrica
- Postprocesado (pre inicialización)
- Inicialización
- Postprocesado (post inicialización)
- Bean listo para uso
- Destrucción

Hay diferentes formas de crear un Bean en Spring:

- Bean simples: sin atributos
- Bean con inyección por constructor: pasando los atributos por constructor
- Bean con referencias de objeto de constructores: cuando pasamos un bean como atributo del constructor de otro.
- Bean con inyección de propiedades: cuando en vez del método constructor utilizamos setters de atributos.
    - Con valores simples: Enteros, reales, Cadenas…
    - Con valores complejos:
        - Por referencia de otro objeto: pasando un bean id al método set.
        - Colecciones de datos: listas, arrays, maps…
- Bean con valor nulo: cuando necesitamos pasar un valor nulo.

```
//tenemos un constructor con un String y un objeto Sombrero
public Persona(String nombre, Sombrero sombrero){
   this.nombre = nombre;
   this.sombrero = sombrero;
}
```

```
<bean id="sombrero" class="info.hcosta.ejemplo.Sombrero" />

<bean id="persona" class="info.hcosta.ejemplo.Persona">
   <constructor-arg value="Juan Carlos" />
   <constructor-arg ref="sombrero" />
</bean>
```

Figura 2-11 Bean con referencias de objeto de constructores

```
//tenemos un setter con un String
public setNombre(String nombre){
   this.nombre = nombre;
}

//tenemos otro setter con un Sombrero
public setSombrero(Sombrero sombrero){
   this.sombrero= sombrero;
}
```

```
<bean id="sombrero" class="info.hcosta.ejemplo.Sombrero" />

<bean id="persona" class="info.hcosta.ejemplo.Persona">
   <property name="nombre" value="Juan Carlos" />
   <property name="sombrero" ref="sombrero" />
</bean>
```

Figura 2-12 Bean con inyección de propiedades por referencia de otro objeto.

2.4.7.2 Uso en nuestro proyecto

Los Spring Beans los hemos utilizado en nuestro proyecto para definir los elementos DAO relacionados con los eventos, locales y propietarios y el dataSource.

2.4.8 Patrón DAO

2.4.8.1 Introducción

DAO es un patrón de diseño software cuyo acrónimo es Data Access Object (Objeto de Acceso a Datos, *en español*). Pertenece al catálogo de Core J2EE Patterns de Java, por lo que no es exclusivo de Spring, pero es uno de los patrones de diseño para realizar aplicaciones con este framework que hagan uso de bases de datos.

En una aplicación J2EE necesitamos acceder a datos, ya sea por persistencia, jdbc, LDAP, etc. Esto puede suponer un problema, pues la forma de acceder a los datos depende del fabricante y del tipo de almacenamiento al que estamos accediendo.

Los componentes de nuestra aplicación deben ser transparentes en la medida de lo posible al actual sistema de almacenamiento de datos para permitir migraciones entre distintos fabricantes, distintos tipos de almacenamiento y diferentes almacenamientos de datos. En el caso de que en un momento dado queramos cambiar el motor de almacenamiento de datos, seguir este modelo nos facilitará la tarea.

El patrón DAO hace uso, como su nombre indica, de un Objeto de Acceso a Datos para abstraer y encapsular el acceso a dichos datos. Un objeto DAO obtiene y guarda los datos donde almacenamos los datos, manejando la conexión con el mismo.

Un DAO siempre realiza operaciones atómicas en la base de datos, nunca son necesarias las transacciones. Claros ejemplos de esto serían búsquedas por una clave, creación, actualización y borrado de un registro, obtener todos los registros y cualquier otra operación que vayamos a realizar muy a menudo.

Normalmente se crea un DAO por cada Objeto que usemos en nuestra aplicación. Dicho objeto puede ser muchas cosas, según estemos usando simplemente jdbc o un framework de persistencia (Hibernate).

2.4.8.2 Uso en nuestro proyecto

Nosotros hemos utilizado este patrón con el fin de generar un objeto especial denominado DAO, para cada una de las clases principales de nuestra aplicación (propietarios, locales y eventos) y así realizar operaciones sobre los mismos con repercusión en la base de datos, ya que como se ha comentado en el punto anterior, actualizamos, creamos y eliminamos objetos asiduamente. Esto además, facilita la migración a otro tipo de base de datos, ya que por ejemplo nosotros estamos haciendo uso de MySQL, pero en el caso de que esto deje de sernos útil y queramos migrar a PostgreSQL lo único que deberíamos hacer es revisar el fichero de consultas SQL asociadas a cada objeto DAO con el fin de revisar la sintaxis SQL propia (ya que recordemos que hay ciertas expresiones y sintaxis no compatibles entre MySQL y PostgreSQL). También cabe decir, que deberíamos cambiar la configuración del conector JDBC, ya que el mismo se configura específicamente para una base de datos dada, pero a fin de cuentas, la misma se realiza mediante un fichero database.properties de como mucho, 10 líneas.

2.4.9 JDBC

2.4.9.1 Introducción

Java Database Connectivity, más conocida por sus siglas JDBC es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la que se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

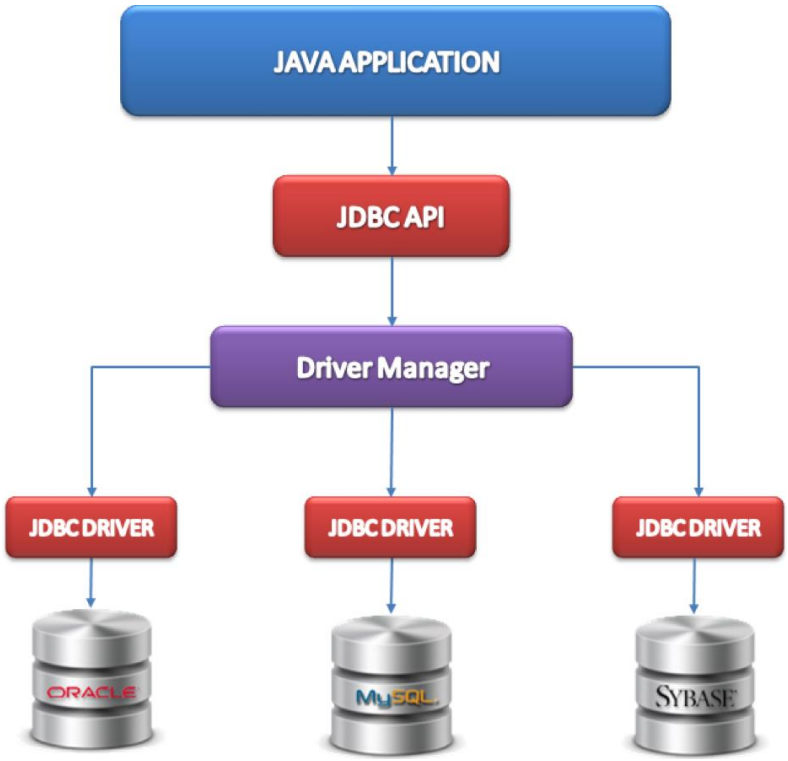


Figura 2-13 Representación gráfica de JDBC.

La API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión: para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

2.4.9.2 Dónde lo hemos utilizado en nuestro proyecto

Nosotros hemos utilizado JDBC, más concretamente el conector JDBC de MySQL para poder conectarnos a la base de datos desde el servicio web REST y así poder realizar todas las operaciones requeridas por los usuarios, relacionadas con datos sobre los mismos, sobre locales o sobre eventos.

2.5 Interfaz web de usuario

2.5.1 Patrón MVC

2.5.1.1 Introducción

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

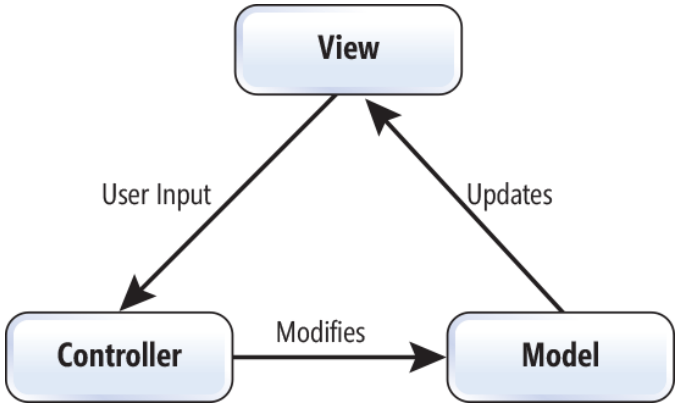


Figura 2-14 Diagrama de interacción en un patrón MVC.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

2.5.1.2 Uso en nuestro proyecto

Nosotros lo utilizamos en la interfaz web de usuario como patrón de diseño. En un comienzo, se tiene un modelo HTML que pasa a ser la vista inicial de usuario, sobre el que se modificarán datos por parte del controlador correspondiente, que en su caso serán o bien ficheros JavaScript o bien el propio usuario dependiendo de la situación, generando cada vez que se modifiquen estos datos, una nueva vista.

2.5.2 HTML5

2.5.2.1 Introducción

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML5 que se sirve de una sintaxis XML (application/xhtml+xml). La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014.

Figura 2-15 Logo gráfico de HTML5.

Al no ser reconocido en versiones antiguas de algunos navegadores web, debido principalmente a la inclusión de nuevas etiquetas con respecto a HTML 4.01, es recomendable actualizar el navegador web a la última versión disponible, con el fin de poder disfrutar de todo el potencial que provee HTML5.

El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

2.5.2.2 Dónde lo hemos utilizado en nuestro proyecto

Nosotros hemos utilizado HTML5 para desarrollar el modelo web, en conjunto con CSS3 sobre el que se representan todos los datos de usuario y que sirve como base para que el usuario tenga una interfaz gráfica para poder interactuar con el servicio web REST, en lugar de tener que hacerlo por otros medios.

2.5.3 CSS3

2.5.3.1 Introducción

CSS (*cascading style sheets*, en inglés) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML). El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que sirven de estándar para los agentes de usuario o navegadores.

Figura 2-16 Logo gráfico de CSS3.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales con la etiqueta <style> o en cada etiqueta particular mediante el atributo "style".

CSS3 es la nueva versión de CSS, retrocompatible con este último, por lo que pueden utilizarse conjuntamente. La principal diferencia es la separación en módulos respecto al CSS original.

2.5.3.2 Dónde lo hemos utilizado en nuestro proyecto

Nosotros hemos utilizado CSS3 para desarrollar el aspecto gráfico del modelo modelo sobre el que se representan todos los datos de usuario en conjunto con HTML5 y que sirve como base para que el usuario tenga una interfaz gráfica para poder interactuar con el servicio web REST, en lugar de tener que hacerlo por otros medios.

2.5.4 JavaScript

2.5.4.1 Introducción

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Figura 2-17 Logo gráfico de JavaScript.

Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque también existe una implementación de JavaScript en el lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF y aplicaciones de escritorio es muy significativo.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

2.5.4.2 jQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción mediante AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es la biblioteca de JavaScript más utilizada del mundo.

Figura 2-18 Logo gráfico de jQuery.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

2.5.4.3 jQuery AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en cualquier aplicación.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, aunque existe la posibilidad de configurar las peticiones como síncronas de tal forma que la interactividad de la página se detiene hasta la espera de la respuesta por parte del servidor.

JavaScript es un lenguaje de programación (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Figura 2-19 Logo gráfico de AJAX en jQuery.

En nuestro caso, usaremos AJAX a través del método $.ajax que es la implementación de dicha tecnología a través de JavaScript, concretamente de la librería jQuery, que es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar una petición de este tipo. Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código.

```
$.ajax({
    url: '/test/PersonSubmit',
    type: 'post',
    dataType: 'json',
    success: function (data) {
        $('#target').html(data.msg);
    },
    data: person
});
```

Figura 2-20 Sintaxis típica de una consulta AJAX en jQuery.

2.5.4.4 jQuery Validation Plugin

El jQuery Validation Plugin es un plugin de jQuery cuyo fin principal es acometer la validación de formularios HTML en el lado del cliente, ofreciendo una gran cantidad de opciones personalizables. Por defecto, el idioma que incluye y que por lo tanto muestra en los mensajes de alerta al realizar la validación de un formulario es el inglés, aunque pueden configurarse otros.

Figura 2-21 Logo gráfico de jQuery Validation Plugin.

2.5.4.5 Google Maps JavaScript API

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle con Google Street View.

Fue anunciado por primera vez en Google Blog el 8 de febrero de 2005. Originalmente soportaría solo a los usuarios de Internet Explorer y Mozilla Firefox, pero el soporte para Opera y Safari fue añadido el 25 de febrero de 2005. El software estuvo en fase beta durante seis meses, antes de convertirse en parte de Google Local, el 6 de octubre de 2005.

Figura 2-22 Logo gráfico de Google Maps.

Como en las aplicaciones web de Google, se usan un gran número de archivos Javascript para crear Google Maps. Como el usuario puede mover el mapa, la visualización del mismo se descarga desde el servidor de Google. Cuando un usuario busca una localización, la misma es marcada por un indicador en forma de pin, el cual es una imagen PNG transparente imprimida sobre el mapa. Para lograr la conectividad asíncrona con el servidor, Google incluyó el uso de jQuery AJAX dentro de esta aplicación.

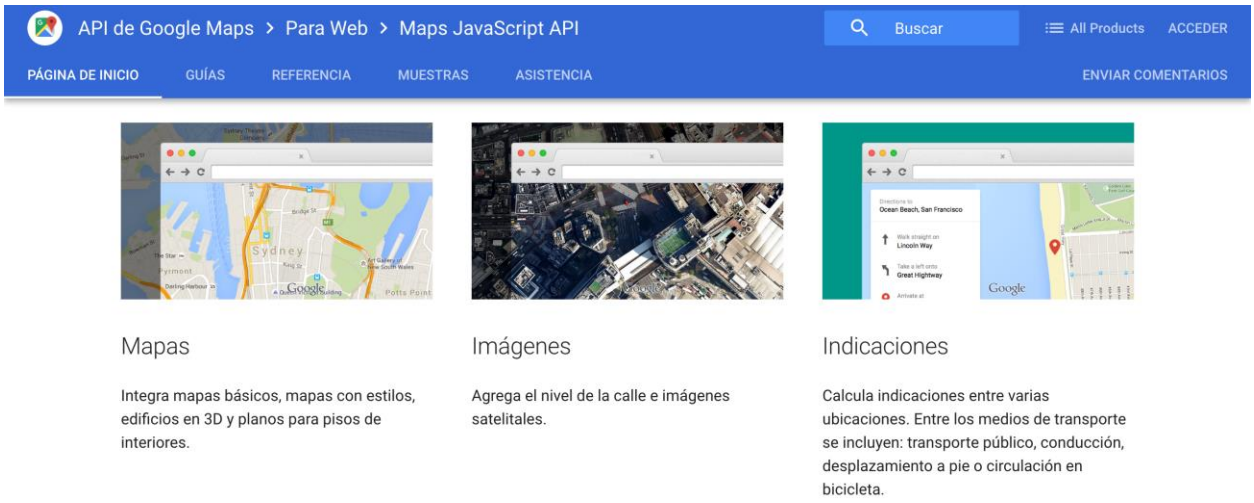


Figura 2-23 Página web principal de la API JavaScript de Google Maps.

La API de Google Maps consiste en una serie de métodos y propiedades que interactúan con los servidores de Google Maps para conseguir el comportamiento de los mapas deseado por el usuario. Para hacer uso de la misma, es necesario estar dado de alta en el servicio Google Developers de Google y obtener una API key que incluiremos en el código JavaScript para poder realizar las funciones relacionadas con los mapas de Google.

Figura 2-24 Página web principal de Google Developers.

Darse de alta en Google Developers es gratuito, pero tiene limitaciones, debido a este mismo aspecto. A pesar de ello, podemos hacer uso de todas las funcionalidades de la API sin coste alguno, como la que utilizamos nosotros en nuestro proyecto, que es la geocodificación: obtenemos las coordenadas a partir de una localización concreta que da el usuario y una vez realiza su confirmación, mostramos dicha localización, a la vez que dichas coordenadas se almacenan en nuestra base de datos. Más adelante entraremos en los detalles relacionados con este aspecto.

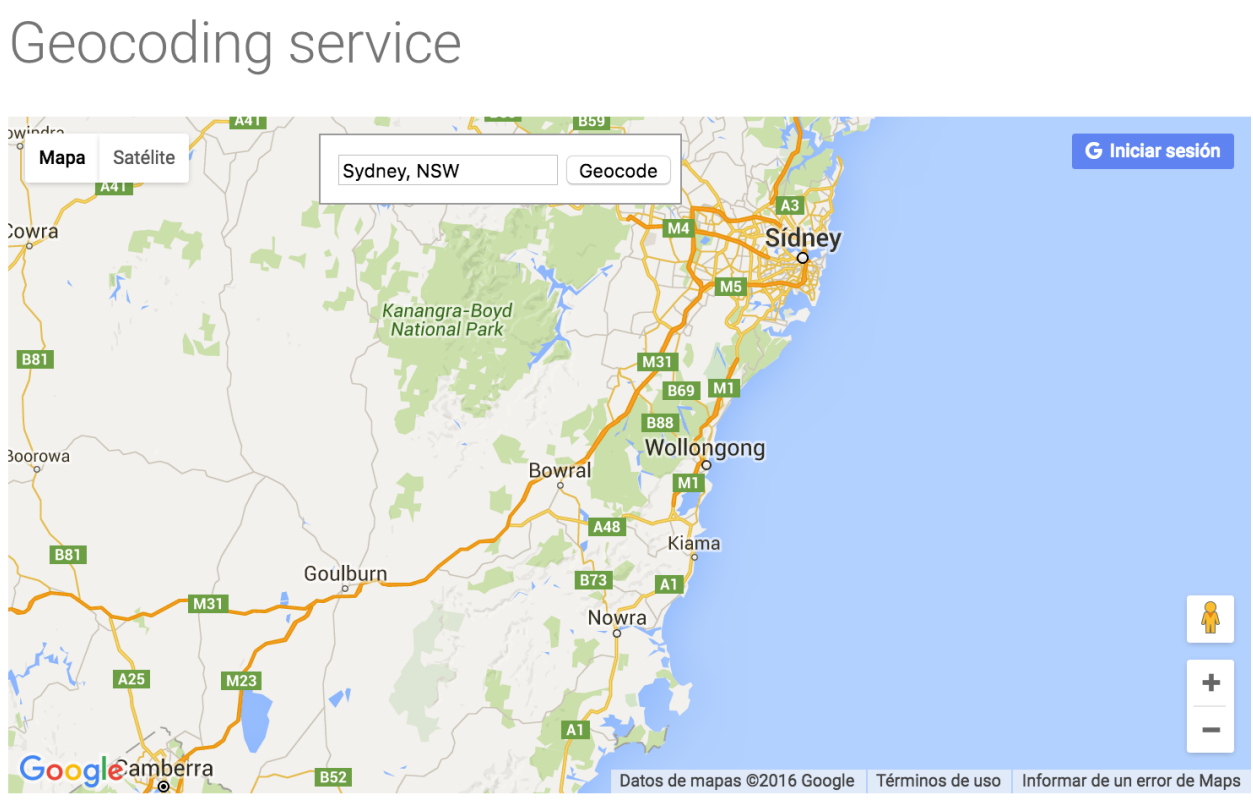


Figura 2-25 Página web que incluye la función de geocodificación de Maps.

2.5.4.6 Uso JavaScript en nuestro proyecto

JavaScript y todas sus librerías,APIs y plugins expuestos lo hemos utilizado, tal y como se observa en la Figura 1-2, para realizar de controladores del modelo en el lado del cliente. Aún así, cada librería, plugin o API realiza una función diferente, que exponemos a continuación:

- **jQuery Validation Plugin**: lo utilizamos para realizar la validación de formularios (es decir, que los datos del formulario cumplan un determinado formato) en el lado del cliente, de tal forma que eliminamos esa carga adicional del servicio web REST Spring.
- **jQuery AJAX**: lo utilizamos para obtener y enviar datos al servicio web REST Spring en formato JSON, haciendo uso de los verbos POST, GET y DELETE (no hemos utilizado PUT por problemas de compatibilidad con algunos navegadores web y para evitar problemas de seguridad).
- **Google Maps JavaScript API**: lo utilizamos para ayudar al propietario de un local a localizar el mismo y así poder facilitarnos sus coordenadas mediante geocodificación, que posteriormente almacenaremos en la base de datos MySQL y que en la aplicación Java ayudará a los usuarios a localizar los eventos gracias a la localización de los locales en los que se celebran, que se ha almacenado como previamente se ha especificado.
- **jQuery y JavaScript**: ambos los utilizamos como lenguajes de programación principales de desarrollo de controladores web del lado del cliente. El uso de jQuery viene justificado por el hecho de que manejar el árbol DOM así como sus manejadores de eventos hacen más intuititiva tanto la programación de un controlador, como su lectura a alguien que sea ajeno a JavaScript, podríamos afirmar que es más amigable frente a JavaScript visto desde un punto de vista externo.

3 HERRAMIENTAS DE DESARROLLO

Nunca en mi vida había utilizado una herramienta más con el tiempo. Con trabajo, empeño e ingenio descubrí que no había nada que no pudiera construir en especial, si tenía herramientas.

- Daniel Defoe en su obra Robinson Crusoe, 1719 -

En este capítulo nos centraremos en describir las herramientas de desarrollo que hemos utilizado en la acometida de nuestro proyecto. Si bien es cierto que, tal y como se muestra en la cita que se encuentra sobre este párrafo, las herramientas por sí solas no resuelven problemas que no sabemos hacer, también es cierto que si se conoce la tarea a realizar y qué herramienta debemos utilizar, la cantidad de tiempo dedicada a dicha tarea es menor, aumentando nuestra productividad, efectividad y reduciendo el tiempo de desarrollo. Por todo esto, pensamos que el hecho de escoger unas buenas herramientas de desarrollo software es fundamental para acometer un proyecto del calibre que se espera de una Escuela Técnica Superior de Ingeniería.

3.1 Spring Tool Suite

3.1.1 Introducción

Spring Tool Suite, también conocido simplemente como STS, proporciona un entorno ready-to-use para implementar, depurar, ejecutar y desplegar las aplicaciones Spring, incluyendo integraciones para Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ. Está desarrollado sobre Eclipse, y normalmente suele actualizarse paralelamente a las últimas versiones de éste.

Figura 3-1 Logo gráfico de Spring Tool Suite.

Spring Tool Suite incluye la edición para desarrolladores de Pivotal tc Server, una versión de Apache Tomcat optimizado para Spring. Con su consola Spring Insight , tc Server Developer Edition ofrece una visión en tiempo real gráfica de los parámetros de rendimiento de aplicaciones que permite a los desarrolladores identificar y diagnosticar los problemas desde sus escritorios.

También soporta el despliegue de aplicaciones tanto en servidores locales, virtuales y en la nube. Es de libre acceso para el desarrollo y uso en operaciones internas sin límite de tiempo, completamente de código abierto y licenciada bajo los términos de la Licencia Pública Eclipse.

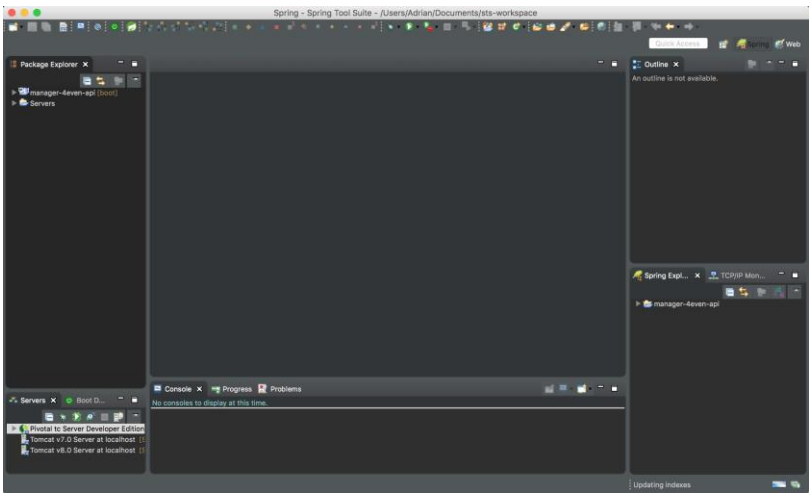


Figura 3-2 Interfaz gráfica de Spring Tool Suite.

3.1.1.1 Uso en nuestro proyecto

Nosotros hemos utilizado Spring Tool Suite en nuestro proyecto como IDE principal para el desarrollo de la aplicación completa: esto es, para el código web del lado del cliente y para el servicio web REST.

Por encima de todo nos ha sido tremendamente útil en el desarrollo del servicio web REST Spring debido al hecho de que te asiste en tiempo real conforme programas de qué funciones puedes hacer uso, cómo puedes hacer uso de las mismas y qué atributos aceptan. Además, incluye una pequeña sugerencia sobre la documentación relacionada de cada función, clase o módulo sobre el que pasemos el ratón.

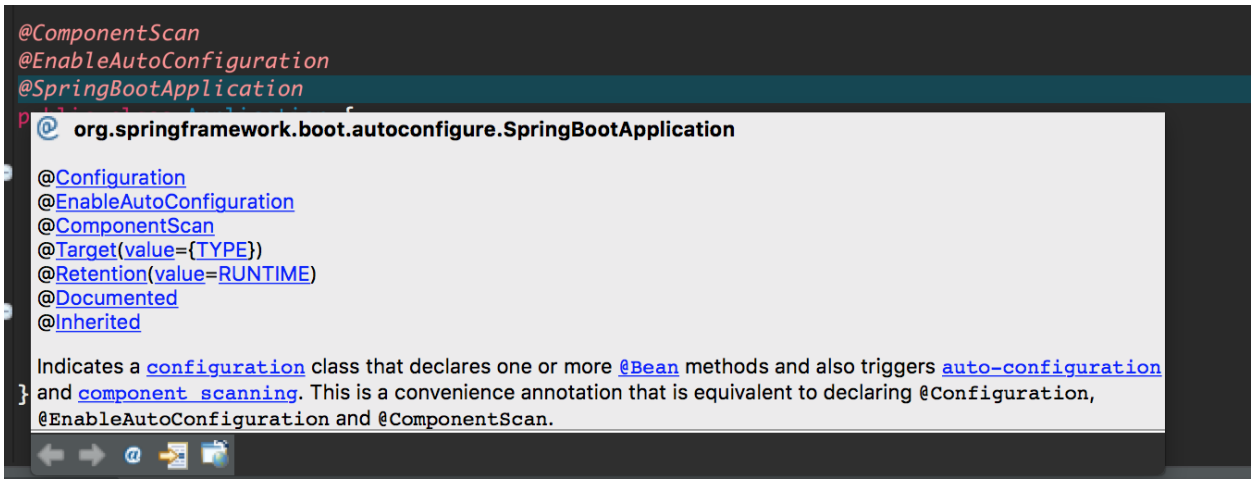


Figura 3-3 Asistencia de programación de Spring Tool Suite.

También ha sido de especial utilidad en cuanto a la validación del código HTML5, ya que lo corrige en tiempo real y comprueba su DOCTYPE, así como otros elementos.

3.2 MySQL Workbench

3.2.1 Introducción

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL. Es el sucesor de DBDesigner 4 de fabFORCE.net, y reemplaza el anterior conjunto de software, MySQL GUI Tools Bundle.

Figura 3-4 Logo gráfico de MySQL Workbench.

3.2.2 Uso en nuestro proyecto

Hemos utilizado MySQL Workbench para desarollar la base de datos MySQL de manera relacional y luego exportarla para su posterior uso con XAMPP.

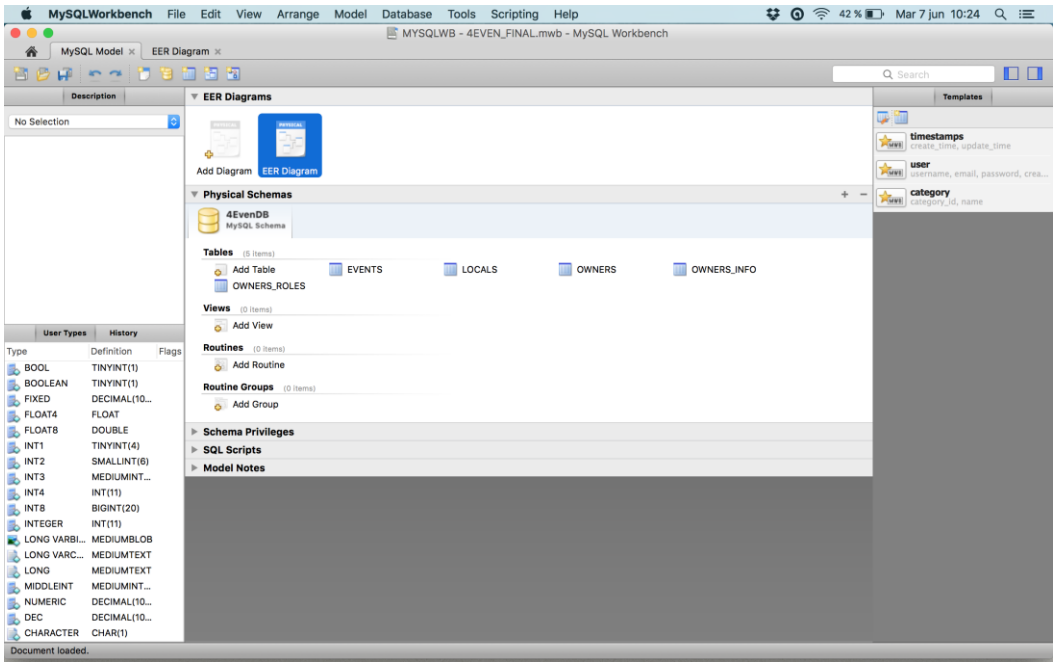


Figura 3-5 Interfaz gráfica de MySQL Workbench.

3.3 XAMPP

3.3.1 Introducción

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. Desde la versión "5.6.15", XAMPP cambió la base de datos de MySQL A MariaDB. El cual es un fork de MySQL con licencia GPL.

Figura 3-6 Logo gráfico de XAMPP.

XAMPP sólo requiere descargar y ejecutar un archivo ZIP, tar , exe o fkl, con unas pequeñas configuraciones en alguno de sus componentes que el servidor Web necesitará. XAMPP se actualiza regularmente para incorporar las últimas versiones de Apache/MySQL/PHP y Perl. También incluye otros módulos como OpenSSL y phpMyAdmin. Para instalar XAMPP se requiere solamente una pequeña fracción del tiempo necesario para descargar y configurar los programas por separado. Puede encontrarse tanto en versión completa, así como en una versión más ligera que es portátil.

3.3.2 Uso en nuestro proyecto

En nuestro proyecto hemos utilizado XAMPP para administrar y ejecutar la base de datos de la aplicación, mediante la herramienta phpMyAdmin incluida en dicha aplicación.

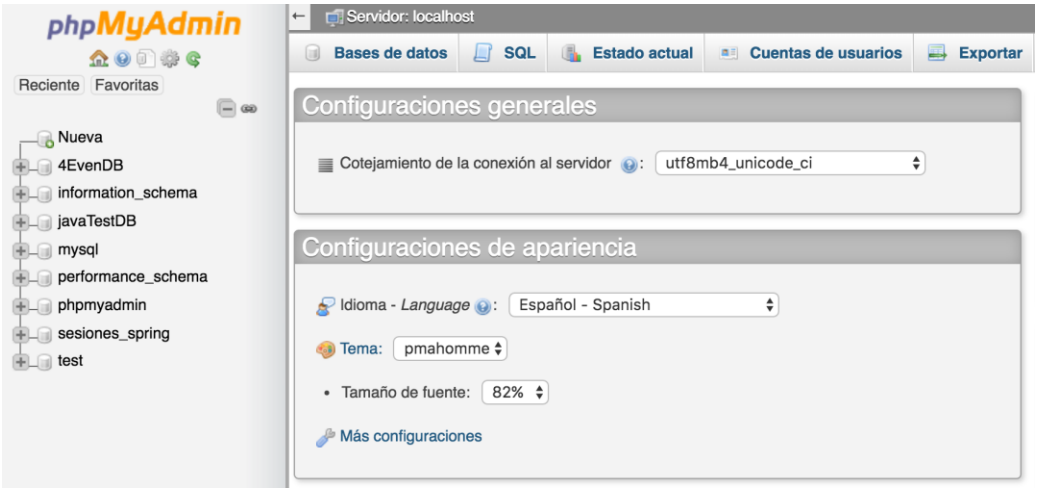


Figura 3-7 Fragmento de interfaz gráfica de phpMyAdmin.

3.4 Postman

3.4.1 Introducción

Postman es una herramienta para el testing de APIs disponible en forma de programa para MacOS y en forma de extensión de Chrome para el resto de sistemas operativos. Es similar a curl, con la diferencia de que dispone de interfaz gráfica, con lo que es mucho más intuitiva. Nos permite construir y gestionar de una forma cómoda nuestras peticiones a servicios REST mediante verbos HTTP (GET, POST, PUT, PATCH, etc.).

Figura 3-8 Logo gráfico de Postman.

Su manejo es muy sencillo ya que solo tenemos que definir la petición que queremos realizar e incluir los datos que queremos enviar (esto último en el caso de verbos como PUT o POST) así como la URL a la que deseamos realizar la petición.

Posteriormente a la petición, muestra la respuesta de la API en el formato estándar de intercambio de la misma ya que es compatible con una gran cantidad de formatos de intercambio de datos, como puede ser XML o JSON.

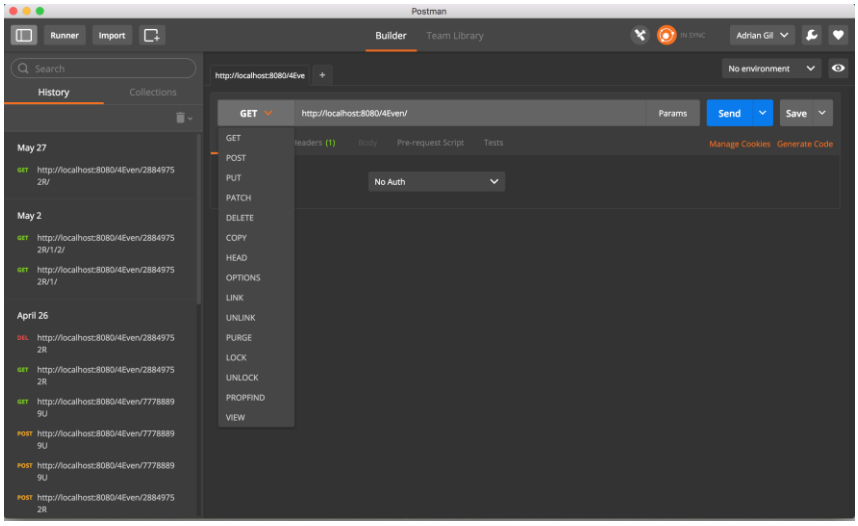


Figura 3-9 Fragmento de interfaz gráfica de Postman.

3.4.2 Uso en nuestro proyecto

Hemos utilizado Postman en nuestro proyecto para realizar la depuración del servicio web REST y así comprobar que todos los métodos de dicho servicio realizan las funciones asignadas sin errores.

3.5 Navegador web Chrome

3.5.1 Introducción

Google Chrome es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones (frameworks) de código abierto, como el motor de renderizado Blink (bifurcación o fork de WebKit). Está disponible gratuitamente bajo condiciones específicas del software privativo o cerrado. El nombre del navegador deriva del término en inglés usado para el marco de la interfaz gráfica de usuario («chrome»).

Figura 3-10 Logo gráfico de Google Chrome.

Chrome dispone de un soporte para agregar extensiones. Las extensiones en Chrome se encuentran disponibles en modo de tienda web para facilitar su instalación, con más de 12.000 extensiones disponibles hasta el momento. No se requiere reiniciar el navegador para aplicar ninguna extensión, y se instalan automáticamente.

3.5.2 Uso en nuestro proyecto

Nosotros hemos utilizado Google Chrome para la depuración de la presentación de la interfaz web gráfica de usuario y su interoperación con el servicio web, así como la depuración de los controladores de JavaScript y su comportamiento.

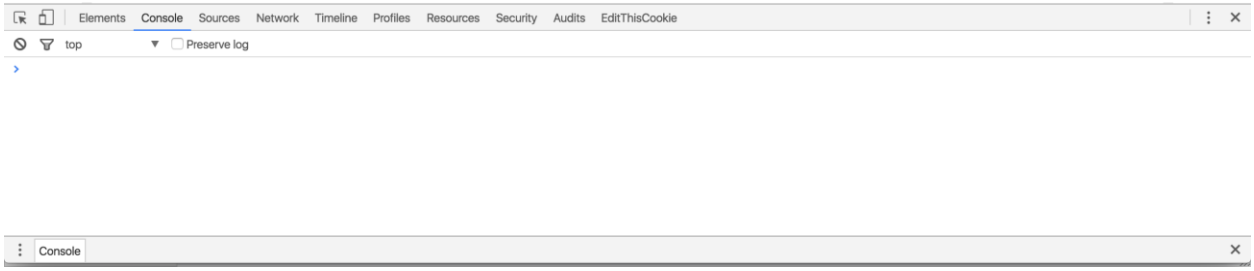


Figura 3-11 Fragmento de la interfaz gráfica de Console, de Google Chrome.

Esto último se ha conseguido activando las herramientas de desarrolladores disponibles en el navegador por defecto. Posteriormente, hemos interactuado según la situación, con las herramientas que nos ofrece, destacando el console, que ha sido la herramienta más utilizada de todas, ya que ahí aparecen o bien los mensajes que en JavaScript escribamos dentro del comando "console.log()" o bien aquellos errores de la página, que normalmente suelen estar relacionados con este lenguaje de programación.

3.6 EditThisCookie

3.6.1 Introducción

EditThisCookie es un gestor de cookies con múltiples funciones, desarrollado por el programador Francesco Capano, debido a que en su opinión, Chrome carecía de un buen gestor de cookies que abarcara todas las funciones posibles relacionadas con cookies y que su aprendizaje fuera sencillo.

Figura 3-12 Logo gráfico de EditThisCookie.

Esta extensión dispone de varias funciones, de entre las que destacamos:

- Eliminar/Editar/Añadir/Buscar una cookie.
- Proteger/Bloquear una cookie mediante filtros.
- Importar/Exportar cookies en formato JSON.

3.6.2 Uso en nuestro proyecto

En nuestro proyecto lo hemos utilizado con el fin de poder depurar las cookies de sesión y protección CSRF de la aplicación web.

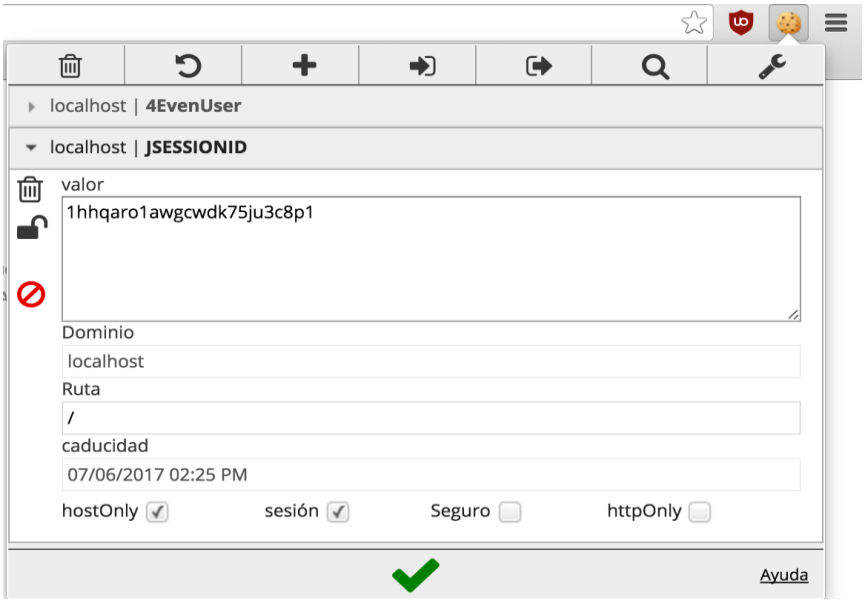


Figura 3-13 Fragmento de interfaz gráfica de EditThisCookie.

4 ARQUITECTURA Y ANÁLISIS

En general, cuando se soluciona un problema nuevo, es conveniente desarrollar primero un modelo simplificado para obtener una idea general de la solución.

- Katsuhiko Ogata -

En este capítulo nos centraremos en describir la arquitectura de la aplicación que hemos realizado como objetivo de nuestro proyecto, la cual también analizaremos a través de una serie de diagramas basados en el estándar UML. Si bien es cierto, tal y como nos aparece en la cita superior de este párrafo, que nuestra aproximación al desarrollo de un servicio web REST con Spring Framework es un modelo sencillo, es necesario su análisis para poder obtener la idea general del funcionamiento del mismo, por lo que esta tarea no es ni mucho menos trivial o prescindible.

4.1 Diagramas de casos de uso

4.1.1 Identificación de los actores del sistema

En los diagramas de casos de uso disponemos de dos actores:

- **Usuario Autenticado (UA)**: Es el usuario que dispone de un perfil creado en el sistema de gestión de eventos y que una vez introduce sus credenciales de usuario (correo y contraseña) puede acceder a todas las funciones que le ofrece el sistema de gestión de eventos. Lo abreviaremos como UA en algunos lugares de los diagramas, con el fin de aumentar el espacio útil dentro de los diagramas.
- **Usuario No Autenticado (UNA)**: Es el usuario que no dispone de un perfil creado en el sistema de gestión de eventos y debe o bien autenticarse o bien darse de alta en el servicio, en el caso de que no tenga cuenta, con el fin de acceder a todas las funciones que le ofrece el sistema de gestión de eventos. Lo abreviaremos como UNA en algunos lugares de los diagramas, con el fin de aumentar el espacio útil dentro de los diagramas.

También haremos uso de la abreviación **SGE** para referirnos al **Sistema de Gestión de Eventos**, así como de la abreviación SAU para referirnos al **Sistema de Atención al Usuario,** con el fin también de ahorrar espacio, sobre todo en la especificación de los casos de uso.

4.1.2 Operaciones de un Usuario No Autenticado

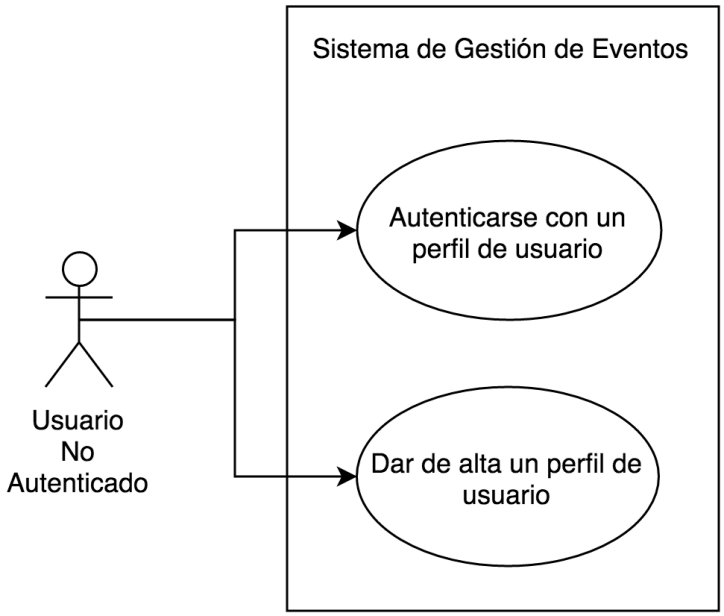


Figura 4-1 Diagrama de casos de uso de operaciones disponibles para un UNA.

Caso de Uso	*Operaciones disponibles para un UNA*
Objetivo	*Realizar operaciones sobre el SGE*
Actores	*Usuario No Autenticado*
Disparador	*Carga la página web del dominio en el que se encuentra alojado el servicio web*
Precondiciones	• *El UNA no debe haberse autenticado previamente ó* • *El UNA no debe disponer de una cuenta de usuario*
Descripción	*El UNA accede a las operaciones disponibles que tiene respecto al SGE.*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UNA se autentica en el SGE (pasa a ser UA)	*El SGE le cede una cookie de sesión y por lo tanto autorización para acceder a sus datos de usuario*
El UNA se da de alta en el SGE	*El SGE registra sus datos y le da la bienvenida*
Cursos alternativos	
Ninguno.	

4.1.3 Operaciones de un Usuario Autenticado

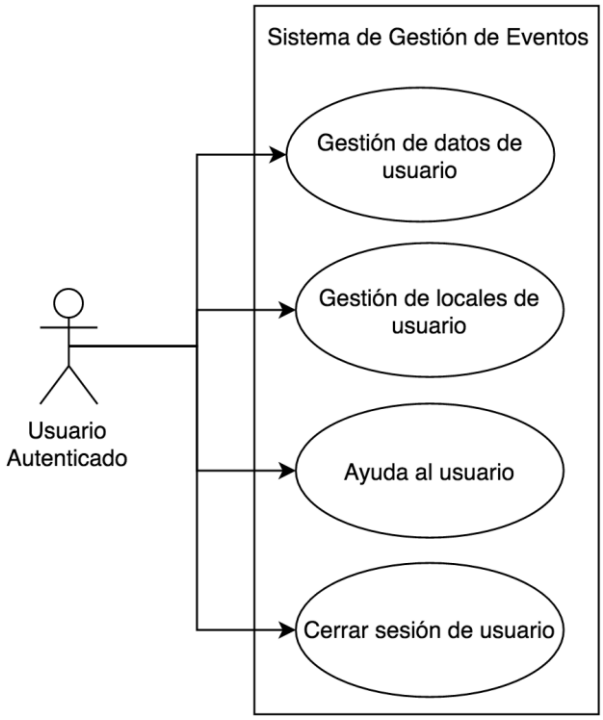


Figura 4-2 Diagrama de casos de uso de operaciones disponibles para un UA

Caso de Uso	*Operaciones disponibles para un UA*
Objetivo	*Realizar operaciones sobre su cuenta del SGE*
Actores	*Usuario Autenticado*
Disparador	*El UA se autentica con éxito en el SGE*
Precondiciones	*El UA debe haberse autenticado correctamente en el SGE*
Descripción	*El UA accede a las operaciones del SGE.*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UA gestiona sus datos de usuario	*El SGE muestra sus datos de usuario y la posibilidad de modificarlos.*
El UA gestiona sus locales	*El SGE muestra los locales registrados por el usuario y las operaciones disponibles respecto a los mismos*
El UA accede a la ayuda de usuario	*El SGE muestra información de ayuda al usuario*
El UA cierra la sesión de usuario (pasa a ser UNA)	*El SGE procede a invalidar la cookie de sesión de usuario*
Cursos alternativos	
Ninguno.	

4.1.3.1 Gestión de datos de usuario

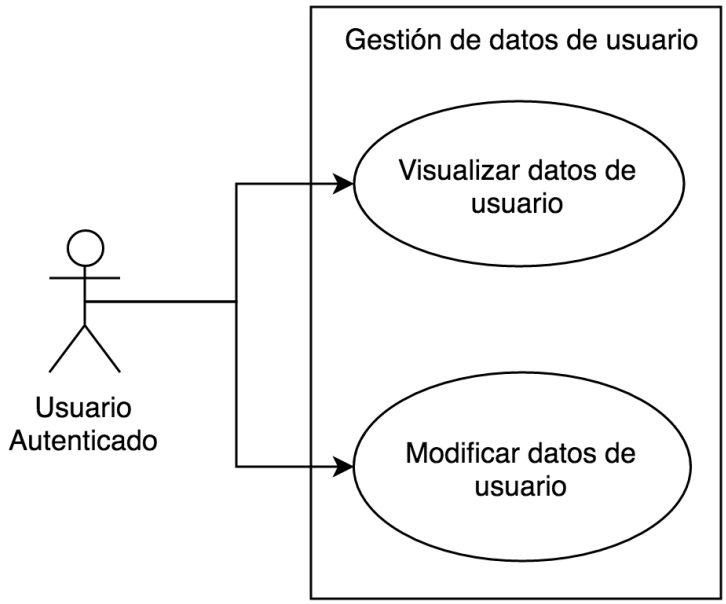


Figura 4-3 Diagrama de casos de uso de gestión de datos de usuario

Caso de Uso	*Gestión de datos de usuario*
Objetivo	*Realizar operaciones sobre la información de la cuenta de usuario*
Actores	*Usuario Autenticado*
Disparador	*El UA realiza click sobre la pestaña Mi Perfil*
Precondiciones	*El UA debe haberse autenticado correctamente en el SGE*
Descripción	*El UA accede a sus datos de usuario, modificándolos o visualizándolos (por defecto).*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UA visualiza sus datos de usuario	*Por defecto, se los muestra*
El UA modifica sus datos de usuario	*El SGE modifica los datos de usuario en la BBDD*
Cursos alternativos	
Ninguno.	

4.1.3.2 Gestión de locales de usuario

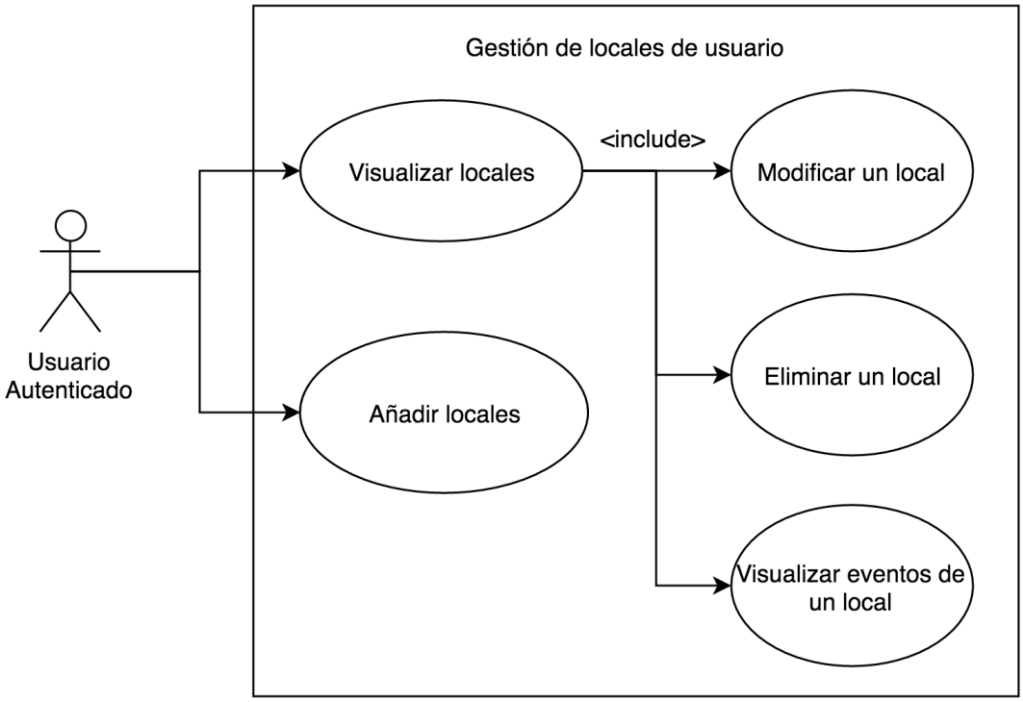


Figura 4-4 Diagrama de casos de uso de gestión de locales de usuario

Caso de Uso	*Gestión de locales de usuario*
Objetivo	*Realizar operaciones sobre la información de los locales de usuario*
Actores	*Usuario Autenticado*
Disparador	*El UA realiza click sobre la pestaña Mis Locales*
Precondiciones	*El UA debe haberse autenticado correctamente en el SGE*
Descripción	*El UA accede a sus locales de usuario, modificándolos o visualizándolos (por defecto).*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UA visualiza sus locales de usuario	*Por defecto, se los muestra*
El UA modifica sus locales de usuario	*El SGE modifica los datos del local de usuario en la BBDD*
El UA añade locales de usuario	*El SGE inserta un nuevo local de usuario en la BBDD*
Cursos alternativos	
Ninguno.	

4.1.3.3 Gestión de eventos de usuario

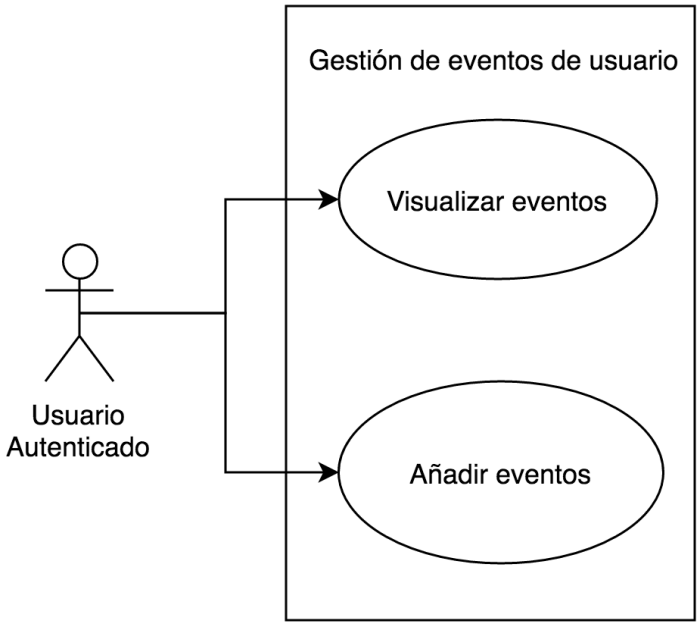


Figura 4-5 Diagrama de casos de uso de gestión de eventos de usuario

Caso de Uso	*Gestión de eventos de usuario*
Objetivo	*Realizar operaciones sobre la información de los eventos de usuario*
Actores	*Usuario Autenticado*
Disparador	*El UA realiza click sobre la pestaña Mis Locales y posteriormente selecciona Ver Eventos de un local concreto*
Precondiciones	*El UA debe haberse autenticado correctamente en el SGE*
Descripción	*El UA accede a sus eventos de un local de usuario, modificándolo o visualizándolo (por defecto).*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UA visualiza sus eventos de usuario de un local	*Por defecto, se los muestra*
El UA modifica sus eventos de usuario de un local	*El SGE modifica los datos del evento de usuario en la BBDD*
El UA añade eventos de usuario de un local	*El SGE inserta un nuevo evento de usuario en la BBDD*
Cursos alternativos	
Ninguno.	

4.1.3.4 Ayuda al usuario

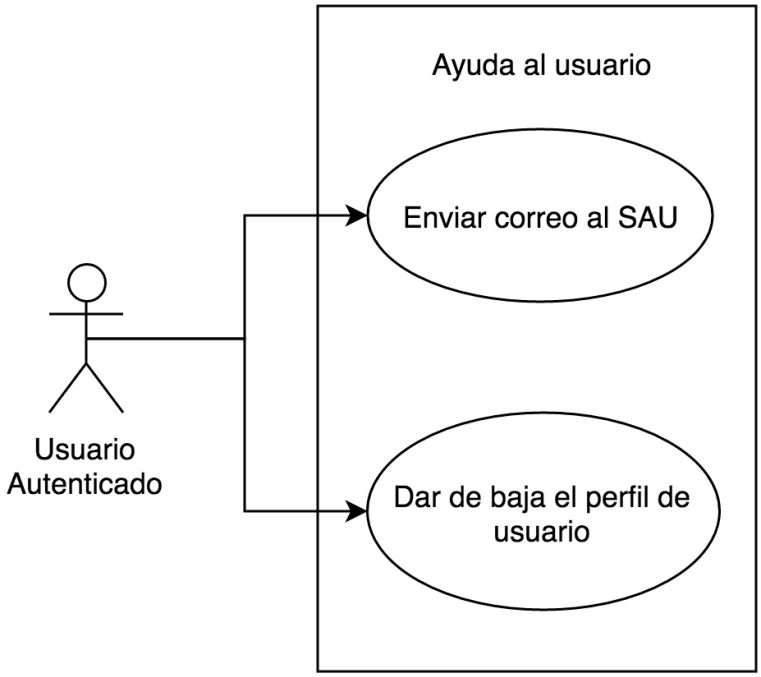


Figura 4-6 Diagrama de casos de uso de ayuda al usuario

Caso de Uso	*Ayuda al usuario*
Objetivo	*Dar de baja el perfil de usuario u obtener ayuda del SGE*
Actores	*Usuario Autenticado*
Disparador	*El UA realiza click sobre la pestaña Ayuda*
Precondiciones	*El UA debe haberse autenticado correctamente en el SGE*
Descripción	*El UA accede a información de asistencia o de eliminación de su perfil de usuario*
Curso normal de los eventos	
Acción de los actores	**Respuesta del sistema**
El UA visualiza ayuda del SGE	*Por defecto, se la muestra*
El UA elimina su cuenta del SGE	*El SGE elimina todos los datos relacionados con el usuario de la BBDD*
Cursos alternativos	
Ninguno.	

4.2 Diagramas de secuencia

4.2.1 Identificación de los elementos de los diagramas

En los diagramas de secuencia disponemos de cinco elementos:

- **Vista Web**: Es lo que ve el usuario en la pantalla de su ordenador, está generada por la acción del Controlador Web sobre el Modelo Web, tal y como se explicó en el capítulo de tecnologías sobre el patrón MVC.
- **Modelo Web**: Es el esqueleto HTML5 sobre el que se representarán datos obtenidos o bien por entrada de usuario o bien por acción directa del Controlador Web.
- **Controlador Web**: Son el conjunto de librerías y funciones escritas en JavaScript que modifican el Modelo Web con el objetivo de definir diferentes Vistas Web de usuario a petición del usuario.
- **Servicio Web**: Es la aplicación desarrollada en Java Spring Framework siguiendo una estructura REST, tal y como explicamos previamente.
- **BBDD**: Es el lugar donde se almacenan los datos relativos a información de usuario, locales y eventos.

Hay que aclarar que sólamente desarrollamos los diagramas de secuencia de mayor importancia en el sistema, ya que el resto son variaciones, dependiendo de los datos que deseemos obtener o enviar al **Servicio Web**.

4.2.2 Autenticación de un usuario

En este diagrama se describe cómo se realiza la autenticación de usuarios en la aplicación.

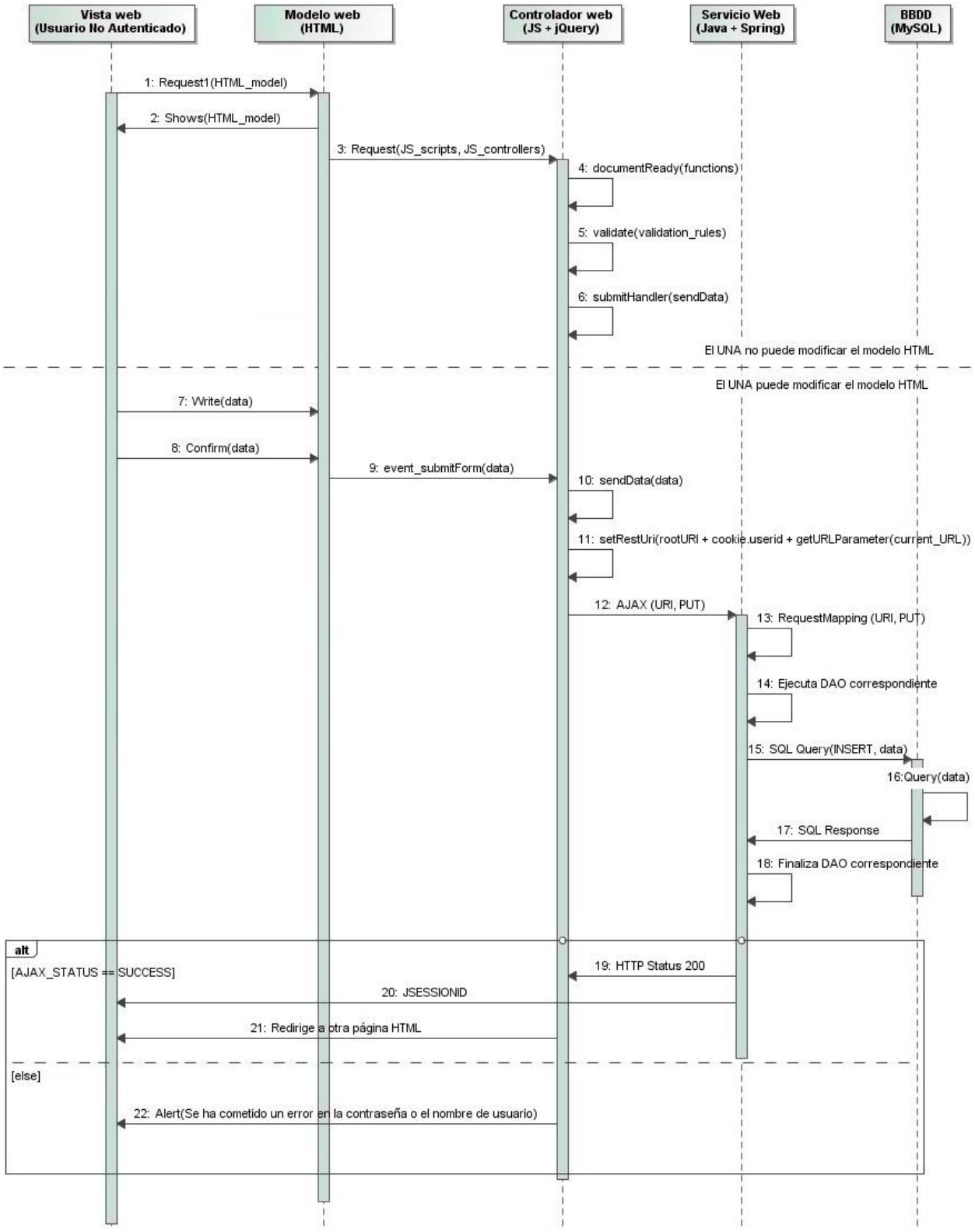


Figura 4-7 Diagrama de secuencia de autenticación de usuario

4.2.3 Obtención de datos

En este diagrama se describe cómo se realiza la obtención de datos en la aplicación. Es válido para el caso de obtención de datos relacionados con eventos, locales o propietarios, ya que lo único que varía es la URI REST y por lo tanto el DAO a ejecutar.

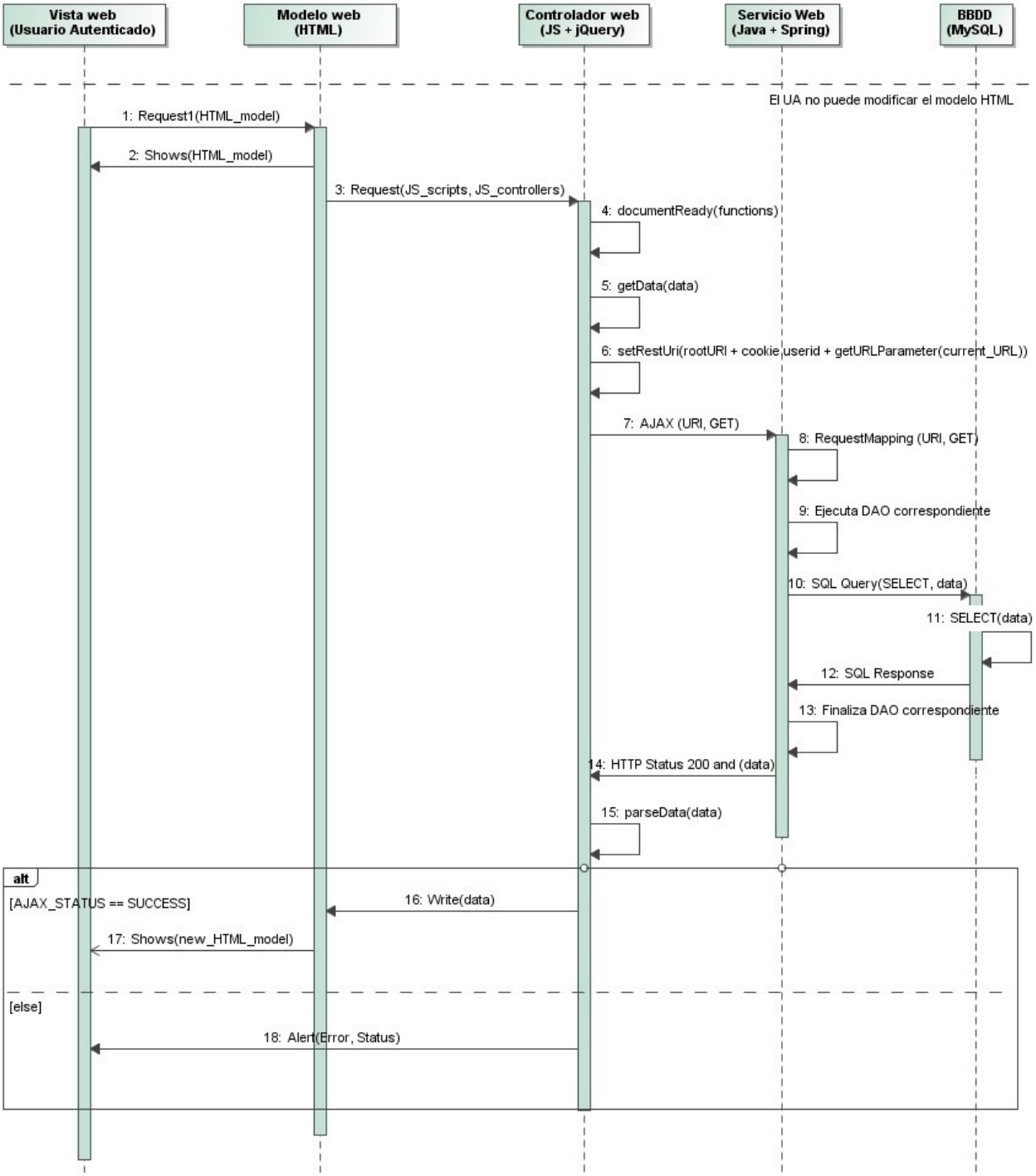


Figura 4-8 Diagrama de secuencia de obtención de datos

4.2.4 Adición de datos

En este diagrama se describe cómo se realiza la adición de datos en la aplicación. Es válido para el caso de adición de datos relacionados con eventos, locales o propietarios, ya que lo único que varía es la URI REST y por lo tanto el DAO a ejecutar.

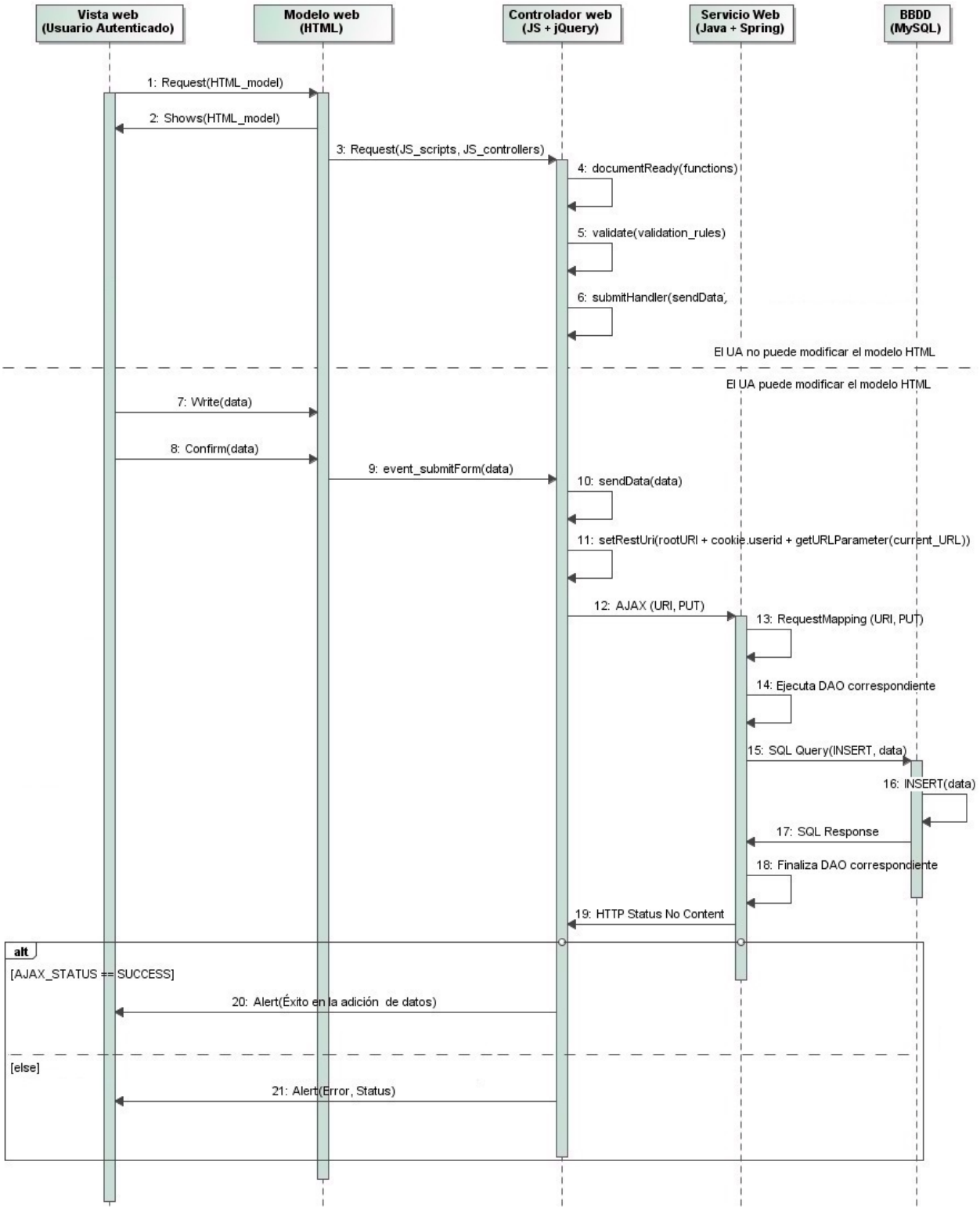


Figura 4-9 Diagrama de secuencia de adición de datos

4.2.5 Modificación de datos

En el caso de que se produzca una modificación de datos en el servicio web, se hace primero una obtención de datos como la de la Figura 5-9 para luego realizar un proceso de adición de datos como el de la Figura 5-9, pero modificando las URIs del servicio REST, así como los DAO, dependiendo de los datos que modifiquemos.

4.3 Diagramas de clases

4.3.1 Clases de gestión de propietarios

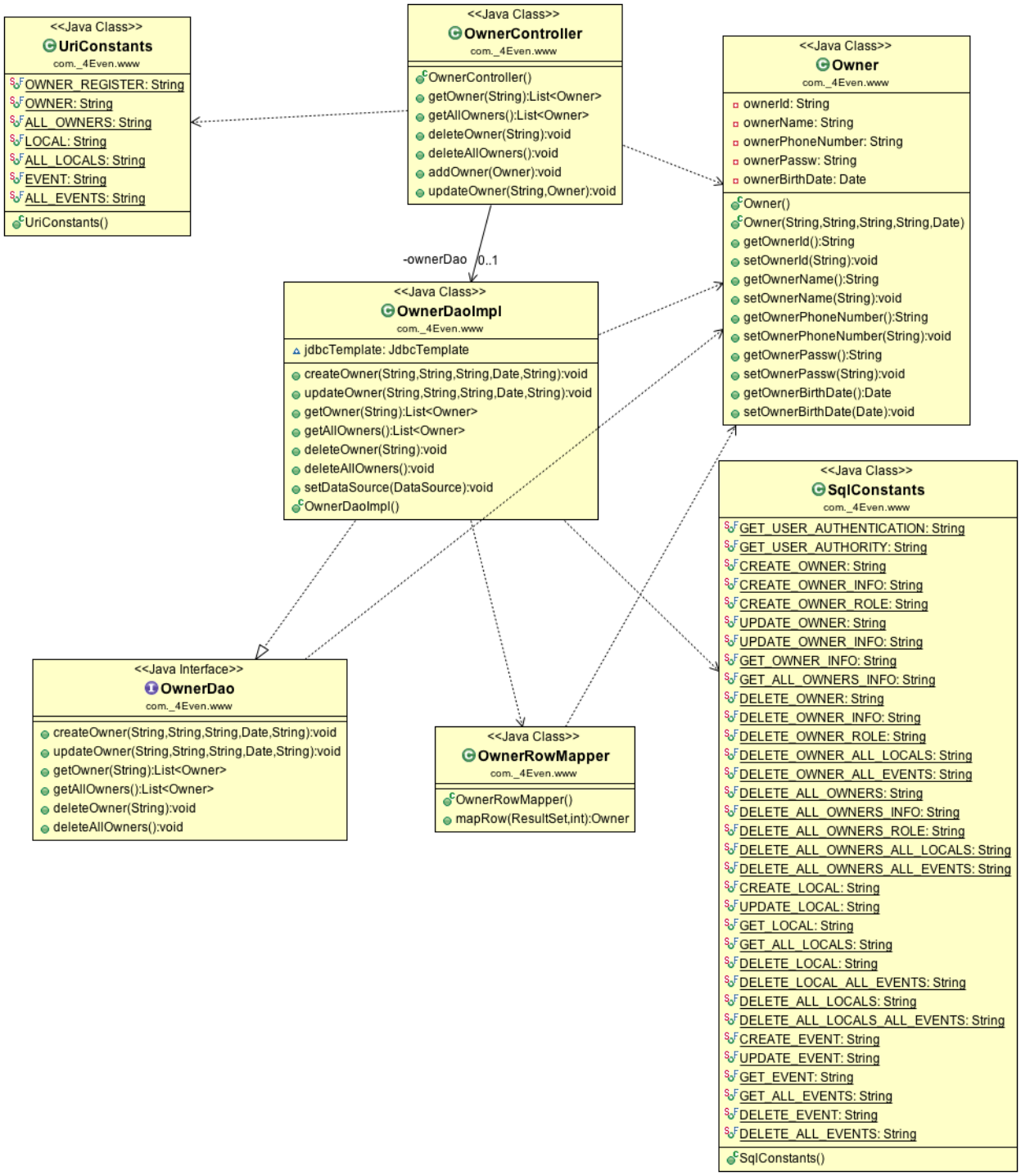


Figura 4-10 Diagrama de clases de gestión de propietarios

4.3.2 Clases de gestión de locales

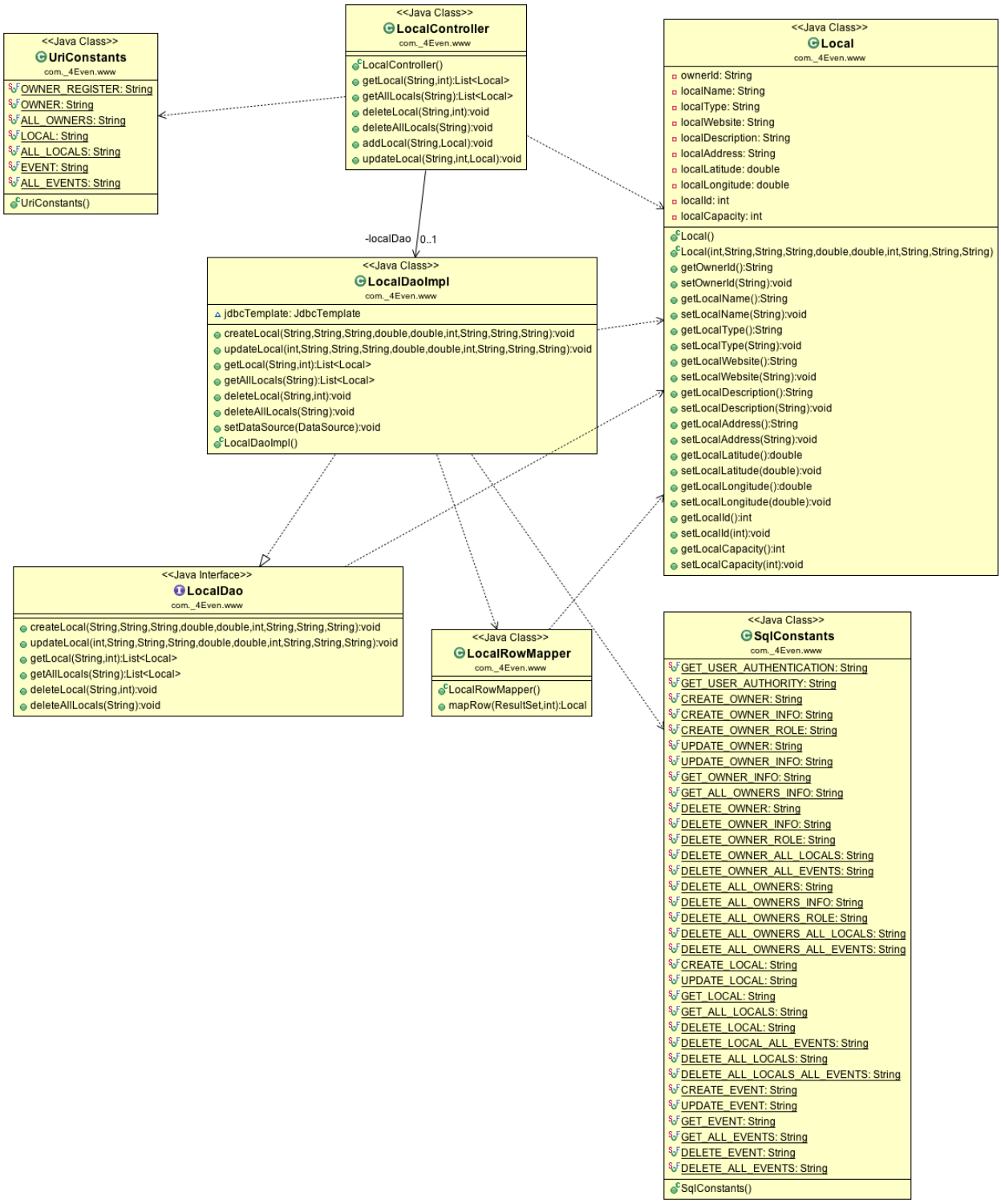


Figura 4-11 Diagrama de clases de gestión de locales

4.3.3 Clases de gestión de eventos

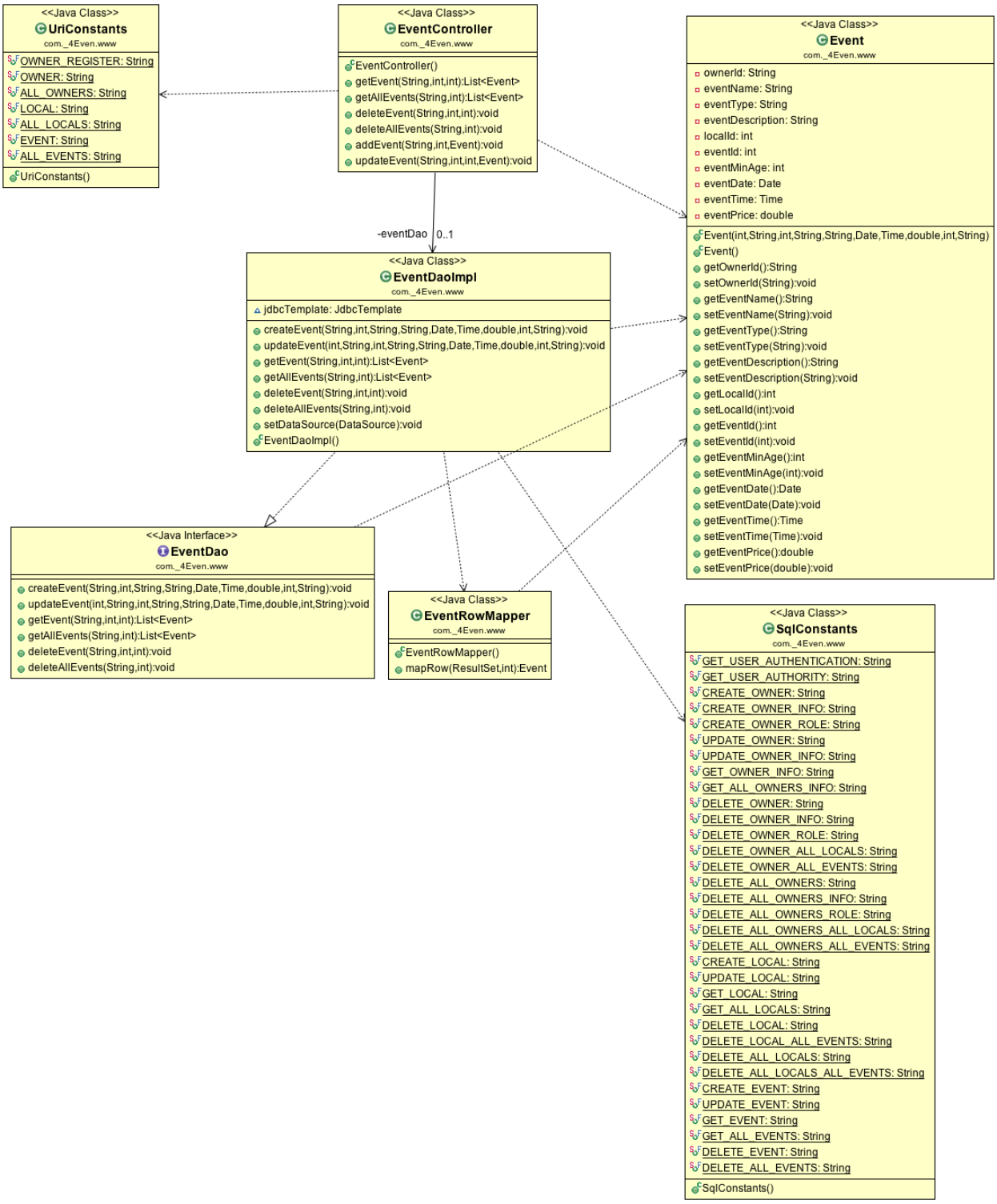


Figura 4-12 Diagrama de clases de gestión de eventos

4.3.4 Clases de gestión de seguridad

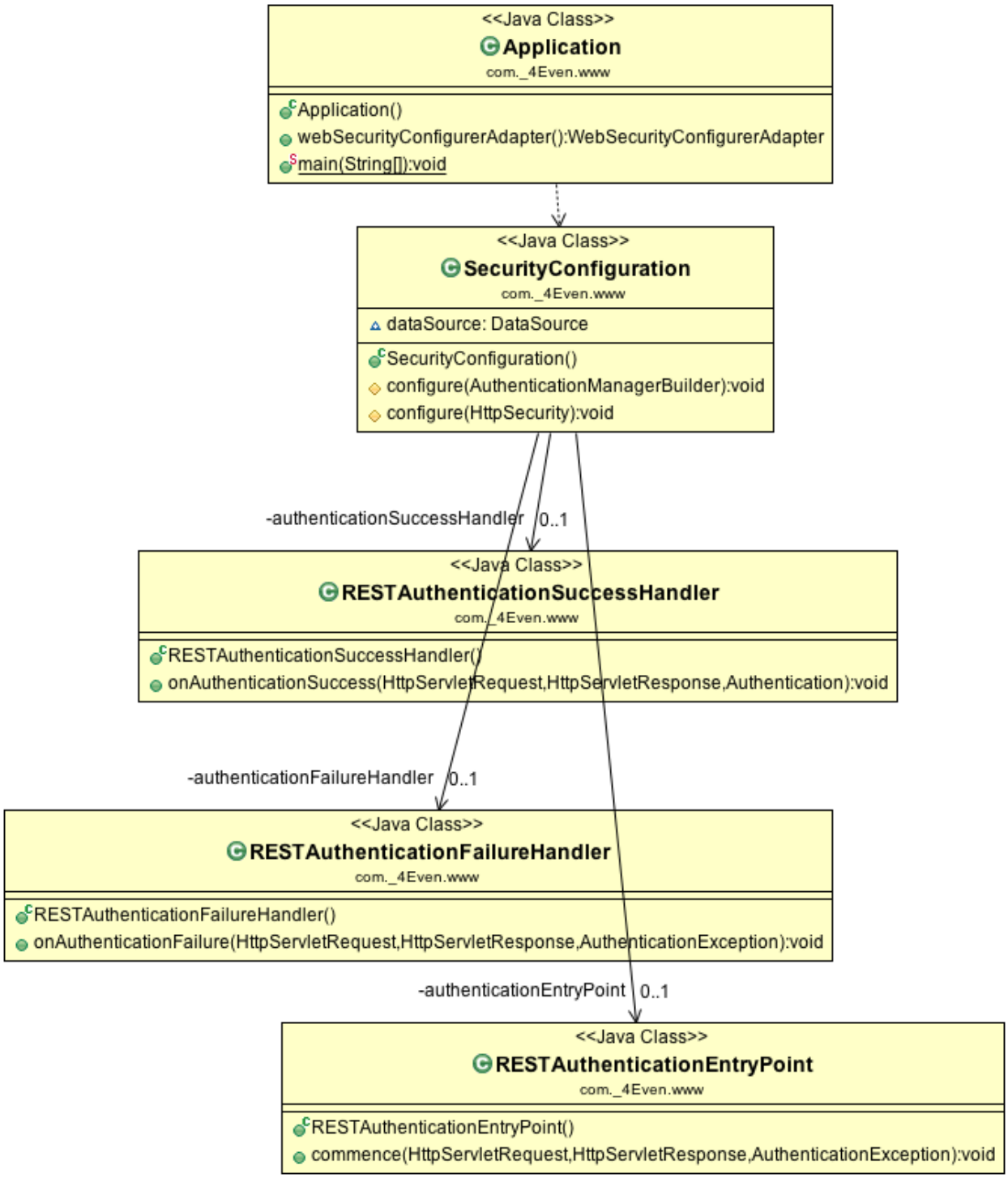


Figura 4-13 Diagrama de clases de gestión de seguridad

4.4 Diagrama EERR de la BBDD

4.4.1 Imagen del diagrama

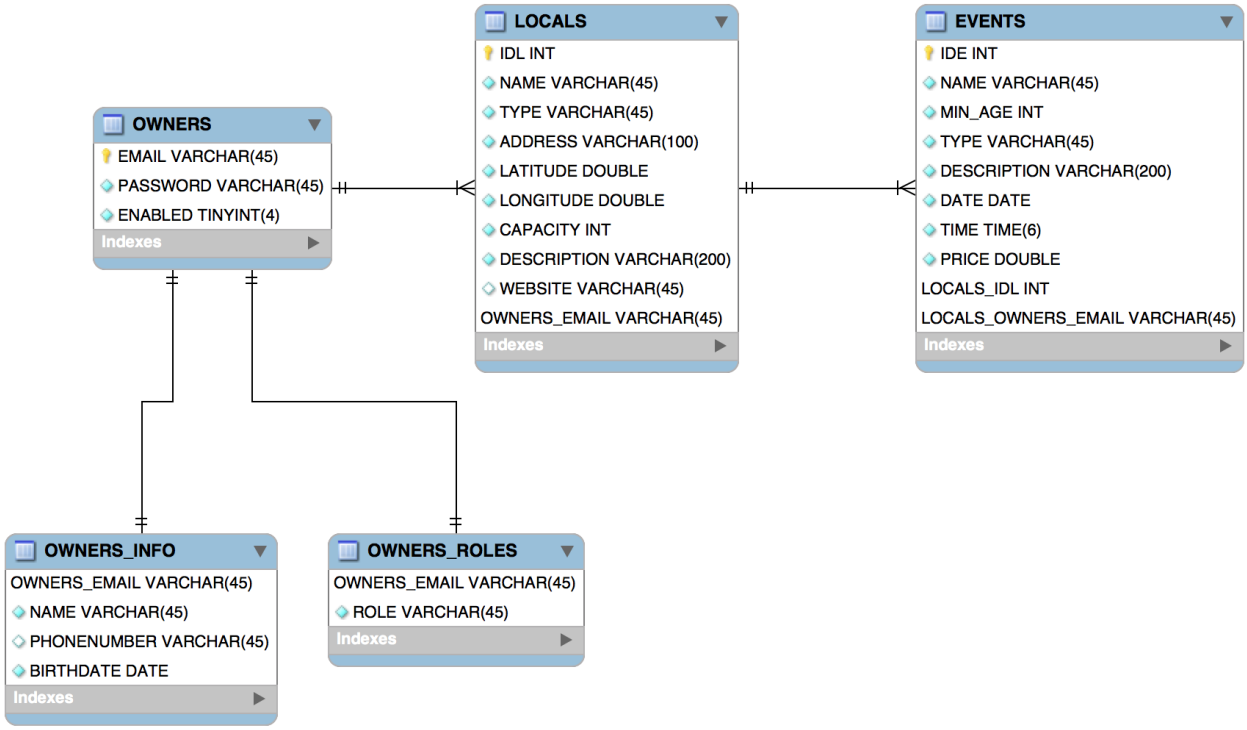


Figura 4-14 Diagrama EERR de la BBDD

4.4.2 Detalle de los elementos del diagrama

4.4.2.1 Tabla OWNERS

Breve Descripción	Es una tabla en la que en cada fila se representa información de acceso al servicio de un usuario del sistema. Un usuario del sistema puede añadir infinitos locales.
EMAIL	*Es el correo electrónico de los propietarios de los locales, que actúa como clave primaria de los datos de los mismos y del sistema en sí, ya que el resto de tablas poseen un campo que es representación como clave externa de este valor. No puede tener valor nulo.*
PASSWORD	*Es la contraseña que utilizan los propietarios para acceder al servicio web. No puede tener valor nulo.*
ENABLED	*Es un valor de carácter Booleano que indica si el usuario está activo o no en el sistema. Lo usa Spring Security para su sistema de autenticación basado en formularios. No puede tener valor nulo.*

4.4.2.2 Tabla OWNERS_INFO

Breve Descripción	Es una tabla en la que en cada fila se representa información general sobre un usuario del sistema.
OWNERS_EMAIL	*Es la clave externa que apunta al campo EMAIL de la tabla OWNERS. No puede tener valor nulo.*
NAME	*Para un propietario, es su nombre junto con sus apellidos. No puede tener valor nulo.*
PHONENUMBER	*Para un propietario, es su número de teléfono. Sí puede tener valor nulo.*
BIRTHDATE	*Para un propietario, es su fecha de nacimiento. No puede tener valor nulo.*

4.4.2.3 Tabla OWNERS_ROLES

Breve Descripción	Es una tabla en la que en cada fila se representa información sobre ROLES de un usuario del sistema. Su existencia está marcada por el uso de Spring Security.
OWNERS_EMAIL	*Es la clave externa que apunta al campo EMAIL de la tabla OWNERS. No puede tener valor nulo.*
NAME	*Para un propietario, es su ROL cuando se autentica en el servicio web. Lo usa Spring Security para su sistema de autorizaciones. No puede tener valor nulo.*

4.4.2.4 Tabla LOCALS

Breve Descripción	Es una tabla en la que en cada fila se representa un local del sistema con toda su información asociada. A un local pueden añadirse infinitos eventos.
IDL	*Es el identificador único de cada local, por lo que es la clave primaria de esta tabla. Es un entero autoincremental. No puede tener valor nulo.*
NAME	*Para un local, es su nombre completo. No puede tener valor nulo.*
TYPE	*Para un local, es su tipo, que es el campo que se utiliza para filtrar en la aplicación Android en la pantalla principal. No puede tener valor nulo.*
ADDRESS	*Para un local, es su dirección, que sólo tiene valor indicativo adicional, ya que tanto la aplicación móvil como el panel de control web hacen uso de los dos siguientes campos para la localización. No puede tener valor nulo.*
LATITUDE	*Para un local, es su latitud, obtenida mediante geocodificación en el panel de control y utilizada por la aplicación Android para la localización de los mismos. No puede tener valor nulo.*
LONGITUDE	*Para un local, es su longitud, obtenida mediante geocodificación en el panel de control y utilizada por la aplicación Android para la localización de los mismos. No puede tener valor nulo.*
CAPACITY	*Para un local, es su aforo máximo disponible. No puede tener valor nulo.*
DESCRIPTION	*Para un local, es una descripción breve del mismo. No puede tener valor nulo.*
WEBSITE	*Para un local, es la página web del mismo. Sí puede tener valor nulo.*
OWNERS_EMAIL	*Es la clave externa que apunta al campo EMAIL de la tabla OWNERS. No puede tener valor nulo.*

4.4.2.5 Tabla EVENTS

Breve Descripción	Es una tabla en la que en cada fila se representa un evento del sistema con toda su información asociada.
IDE	*Es el identificador único de cada evento, por lo que es la clave primaria de esta tabla. Es un entero autoincremental. No puede tener valor nulo.*
NAME	*Para un evento, es su nombre completo. No puede tener valor nulo.*
MIN_AGE	*Para un evento, es la edad mínima de asistencia al mismo. No puede tener valor nulo.*
TYPE	*Para un evento, es su tipo. En este caso, a diferencia del tipo de los locales, sólo tiene valor descriptivo en la aplicación Android.*
DESCRIPTION	*Para un evento, es una descripción breve del mismo. No puede tener valor nulo.*
DATE	*Para un evento, es el día en el que se celebra. No puede tener valor nulo.*
TIME	*Para un evento, es su hora de comienzo. No puede tener valor nulo.*
PRICE	*Para un evento, es el precio de asistencia al mismo. No puede tener valor nulo.*
LOCALS_IDL	*Es la clave externa que apunta al campo IDL de la tabla LOCALS. No puede tener valor nulo.*
LOCALS_OWNERS_EMAIL	*Es la clave externa que apunta al campo EMAIL de la tabla OWNERS. No puede tener valor nulo.*

5 INTERFAZ DE USUARIO Y FUNCIONALIDAD

Aquello que funciona bien es mejor que aquello cuya apariencia es buena. Porque lo que funciona bien permanece.

- Ray Eames -

En este capítulo nos centraremos en describir la funcionalidad de la aplicación de la que es objeto este proyecto, haciendo especial hincapié en la interfaz gráfica de usuario. Consideramos que lo que percibe el usuario es lo más importante de cara a una aplicación como la nuestra, cuya meta principal es que usuarios sin experiencia hagan uso de la misma. Y si bien es cierto que el aspecto gráfico de la aplicación no es especialmente complejo en cuanto a transiciones o efectos gráficos, la simplicidad que ofrece hace que el usuario de la aplicación no se sienta abrumado por la falta o el exceso de información.

5.1 Introducción

Antes que nada, deseamos destacar que debido al carácter panorámico de una página web, se nos hace imposible mostrar de forma completamente fiel la interfaz web gráfica de usuario, de tal forma que los datos sean legibles, por lo que hemos optado por adjuntar los fragmentos más importantes de la misma, de tal forma que podamos reflejar al menos, sus aspectos más importantes y que el lector de este proyecto tenga una idea aproximada de cómo es, sin tener que proceder a la ejecución del servicio web.

Los aspectos gráficos en los que nos centraremos a lo largo de este capítulo, que analizaremos a los largo de pantallas, son los siguientes:

- Inicio de sesión
- Creación de una cuenta
- Gestión de perfil
    - Mostrar perfil
    - Modificar perfil
- Gestión de locales
    - Mostrar locales
    - Modificar locales
    - Gestión de eventos de un local
        - Mostrar eventos de un local
        - Modificar eventos de un local
- Ayuda
    - Enviar un correo
    - Darse de baja
- Cierre de sesión

5.2 Inicio de sesión

Si el usuario teclea en su navegador la dirección web donde se aloja nuestro servicio, la página que aparecerá de inicio es la de la Figura 5-1.

Debido a que es un formulario, hasta que los datos no sean debidamente cumplimentados, el controlador JavaScript no los enviará al servicio web. Además, dicho controlador advertirá al usuario, con mensajes sobre el modelo HTML, sobre qué campos han sido mal completados. Esto es aplicable para todos los formularios de envío de datos de la aplicación.

Hay vdos opciones disponibles:

- Confirmar: Envía los datos y reenvía al usuario a la pantalla de la Figura 5-1. En el caso de que los datos sean correctos en su formato, pero no existan en la base de datos, se advertirá al usuario, con la ventana que se muestra en la Figura 5-2.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-1.

Figura 5-1 Fragmento de la página de inicio de sesión

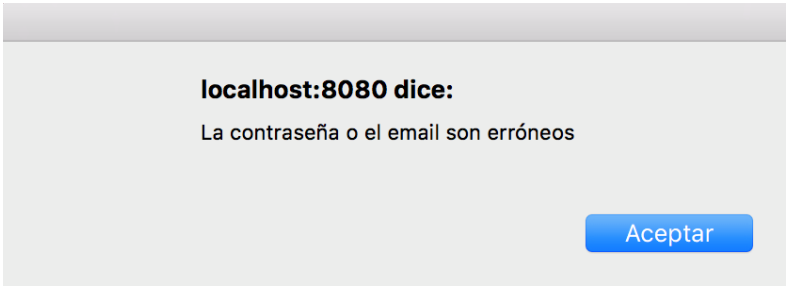


Figura 5-2 Ventana estándar de información

Por otra parte, cabe destacar que una vez se autentique el usuario, aparece en la parte superior derecha del navegador la barra de navegación de la Figura 5-3, que permite al usuario desplazarse entre las principales opciones que ofrece la página web.

Figura 5-3 Barra de navegación de usuario

5.3 Creación de una cuenta

Si el usuario selecciona en su navegador la opción "No tengo una cuenta", la página que aparecerá de inicio es la de la Figura 5-3.

Debido al carácter de formulario, se actúa de una manera similar al caso del apartado 5.2 de este capítulo.

Hay dos opciones disponibles:

- Confirmar: Envía los datos y reenvía al usuario a la pantalla de la Figura 5-1. En el caso de que el registro tenga éxito o no, se informará al usuario con la ventana de la Figura 5-2.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-1.

Figura 5-4 Fragmento de la página de creación de una cuenta

5.4 Gestión de perfil

En este apartado, explicaremos las opciones disponibles por parte del usuario en el caso de que seleccione la opción "MI PERFIL" de la Figura 5-3.

5.4.1 Mostrar perfil

En la Figura 5-5 se observa la página en la que se muestra la información de la cuenta del usuario, también denominado como perfil de usuario. Esta es la opción que se muestra por defecto al pulsar la opción "MI PERFIL" de la Figura 5-3.

No es en sí un formulario, debido a que los campos son de sólo lectura, así que no pueden realizarse modificaciones directamente por parte del usuario sobre esta página.

Por otra parte, sí que pueden modificarse los datos de usuario, en el caso en el que se seleccione la opción "Modificar mi perfil", que redirige al usuario a la página que se muestra en la Figura 5-6.

Figura 5-5 Fragmento de la página de perfil de usuario

5.4.2 Modificar perfil

Si el usuario selecciona en su navegador la opción "Modificar mi perfil" que se muestra en la Figura 5-5, la página que aparecerá es la correspondiente a la Figura 5-6.

La función principal de esta página es actuar como formulario, por lo que su comportamiento es similar el caso que se especificó en el apartado 5.2 de este capítulo.

Hay varias opciones disponibles:

- Confirmar: Envía los datos y redirige al usuario a la página de la Figura 5-5.
- Cancelar: No realiza el envío de datos y redirige al usuario a la página de la Figura 5-5.

Figura 5-6 Fragmento de la página de modificación del perfil de usuario

5.5 Gestión de locales

En este apartado detallaremos las opciones disponibles por parte del usuario en el caso de que seleccione la opción "MIS LOCALES" de la Figura 5-3.

5.5.1 Mostrar locales

En la Figura 5-7 se observa la página en la que se muestra una lista en la que aparecen todos los locales pertenecientes a un usuario, así como un resumen de la información más importante de los mismos. Esto es lo que se muestra por defecto al pulsar la opción "MIS LOCALES" en la Figura 5-3.

Hay varias opciones disponibles:

- Modificar (un local): Nos ofrece la posibilidad de cambiar la información de un local.
- Ver eventos (de un local): Nos ofrece la posibilidad de ver los eventos de un local.
- Eliminar (un local): Nos ofrece la posibilidad de eliminar un local, así como todos los eventos que se celebren en el mismo.
- Añadir un local: Nos ofrece la posibilidad de añadir un local.

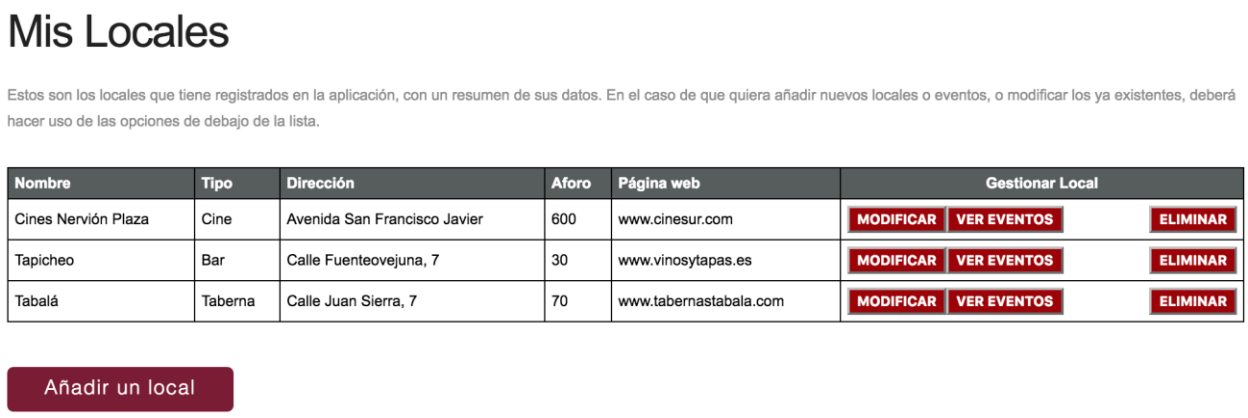


Figura 5-7 Fragmento de la página en la que se muestran los locales de un usuario

5.5.2 Modificar locales

En la Figura 5-8 se muestra la opción disponible en el caso de que seleccionemos "Modificar" sobre un local concreto, de los que aparecen en la lista de la Figura 5-7.

La función principal de esta página es actuar como formulario, por lo que su comportamiento es similar el caso que se especificó en el apartado 5.2 de este capítulo.

Hay dos opciones disponibles:

- Confirmar: Envía los datos del local y redirige al usuario a la pantalla de la Figura 5-7.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-7.

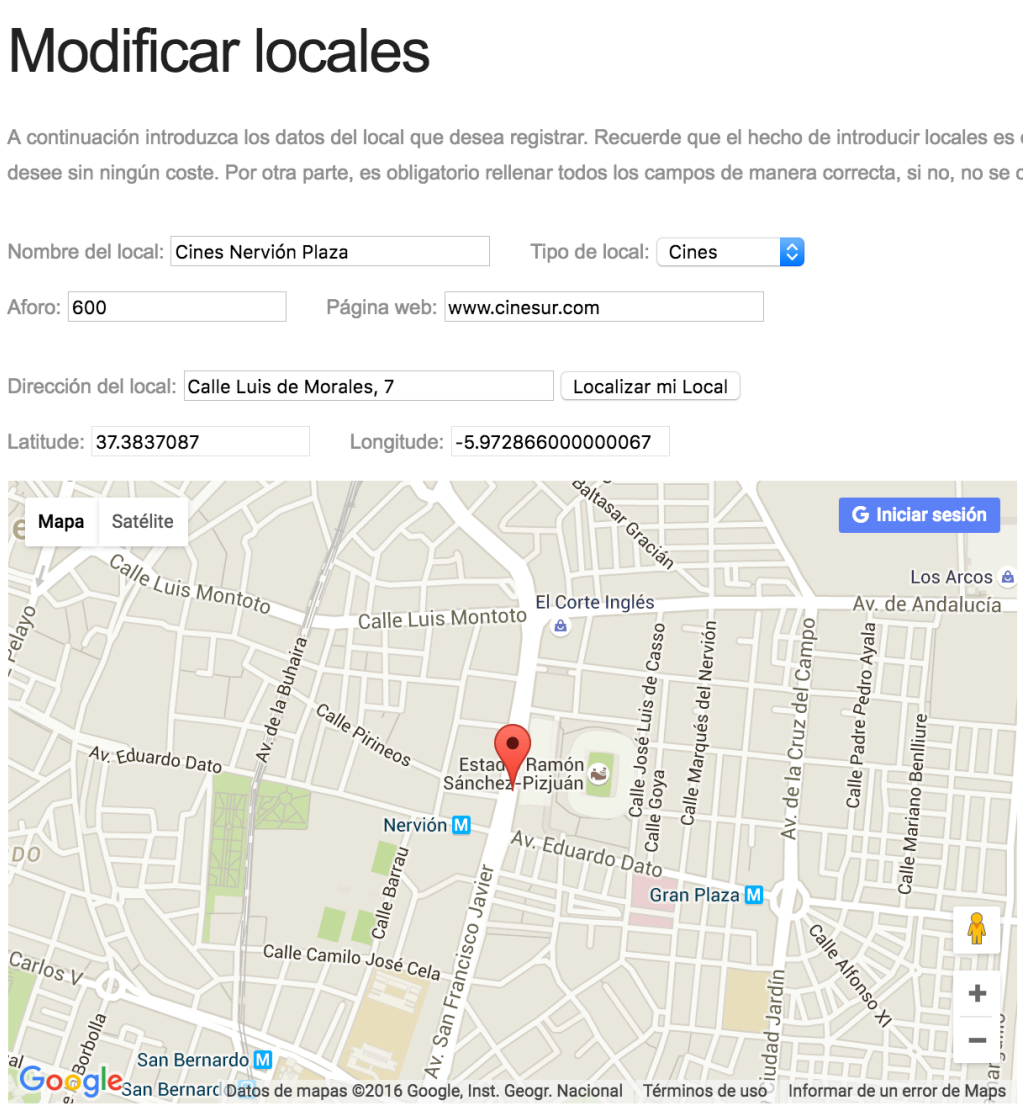


Figura 5-8 Primer fragmento de la página en la que se modifica un local (como se muestra por defecto)

En la Figura 5-9 se observa el comportamiento del mapa si el usuario decide comprobar la correcta localización de su local haciendo uso de la opción Street View disponible en la página web. Por otra parte, en la Figura 5-10 se ve el segundo fragmento correspondiente a la adición de locales. Hemos tenido que separar esta página en dos partes por cuestiones de espacio.

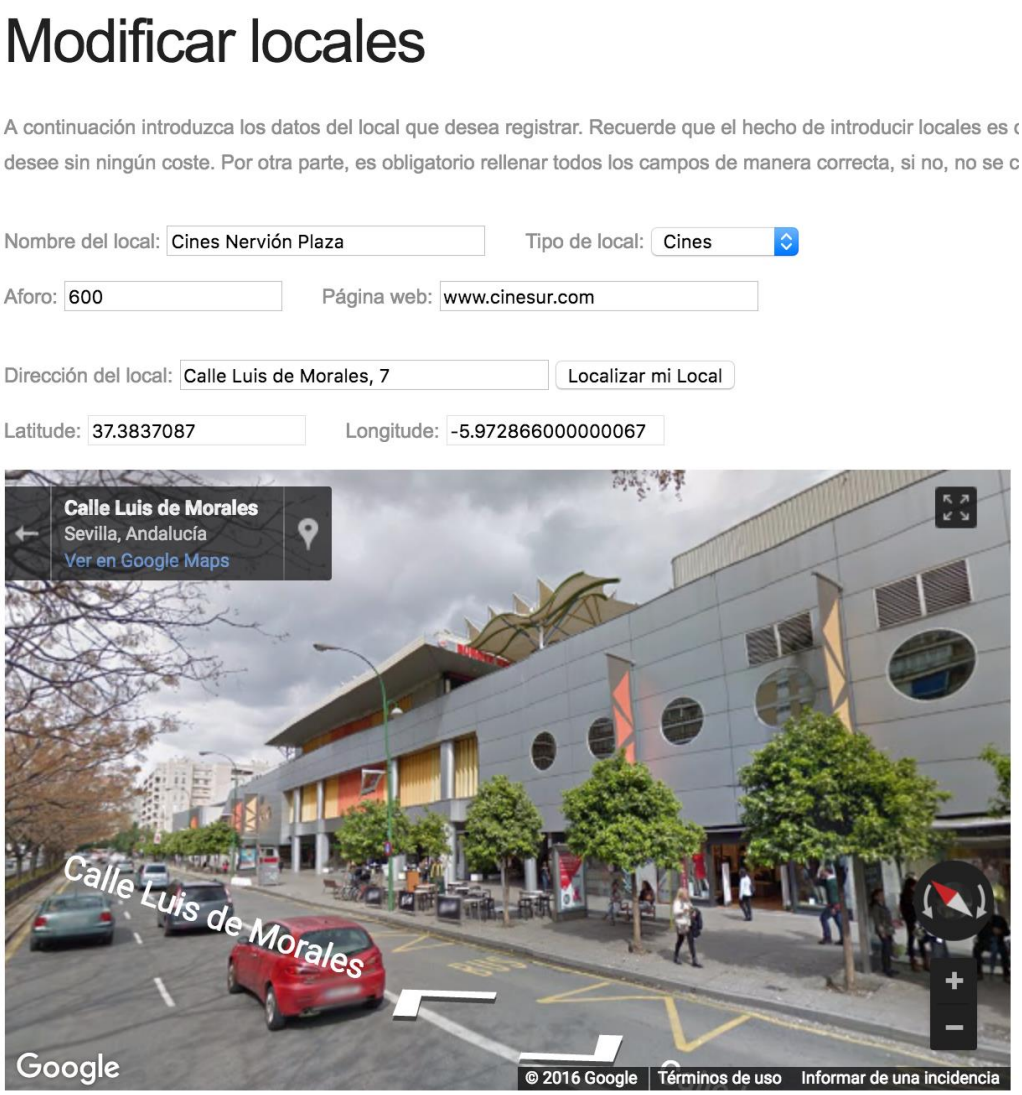


Figura 5-9 Primer fragmento de la página en la que se modifica un local (usando Street View)

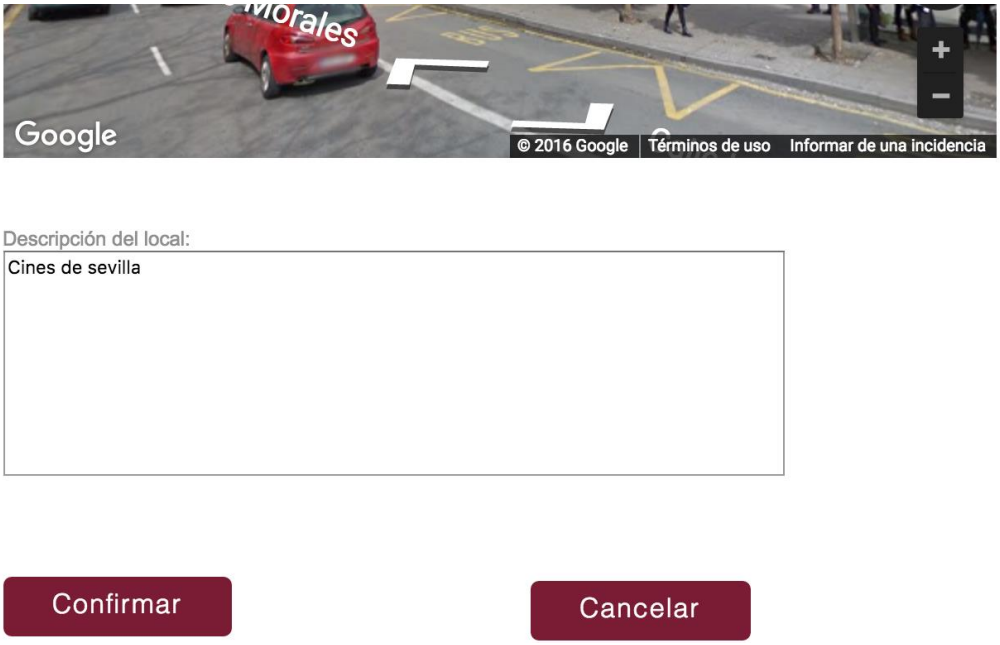


Figura 5-10 Segundo fragmento de la página en la que se modifica un local (usando de Street View)

5.5.3 Añadir locales

En la Figura 5-11 se muestra la opción disponible en el caso de que seleccionemos "Añadir un local" en la Figura 5-7.

La función principal de esta página es actuar como formulario, por lo que su comportamiento es similar el caso que se especificó en el apartado 5.2 de este capítulo.

Hay dos opciones disponibles:

- Confirmar: Envía los datos del y redirige al usuario a la pantalla de la Figura 5-7.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-7.

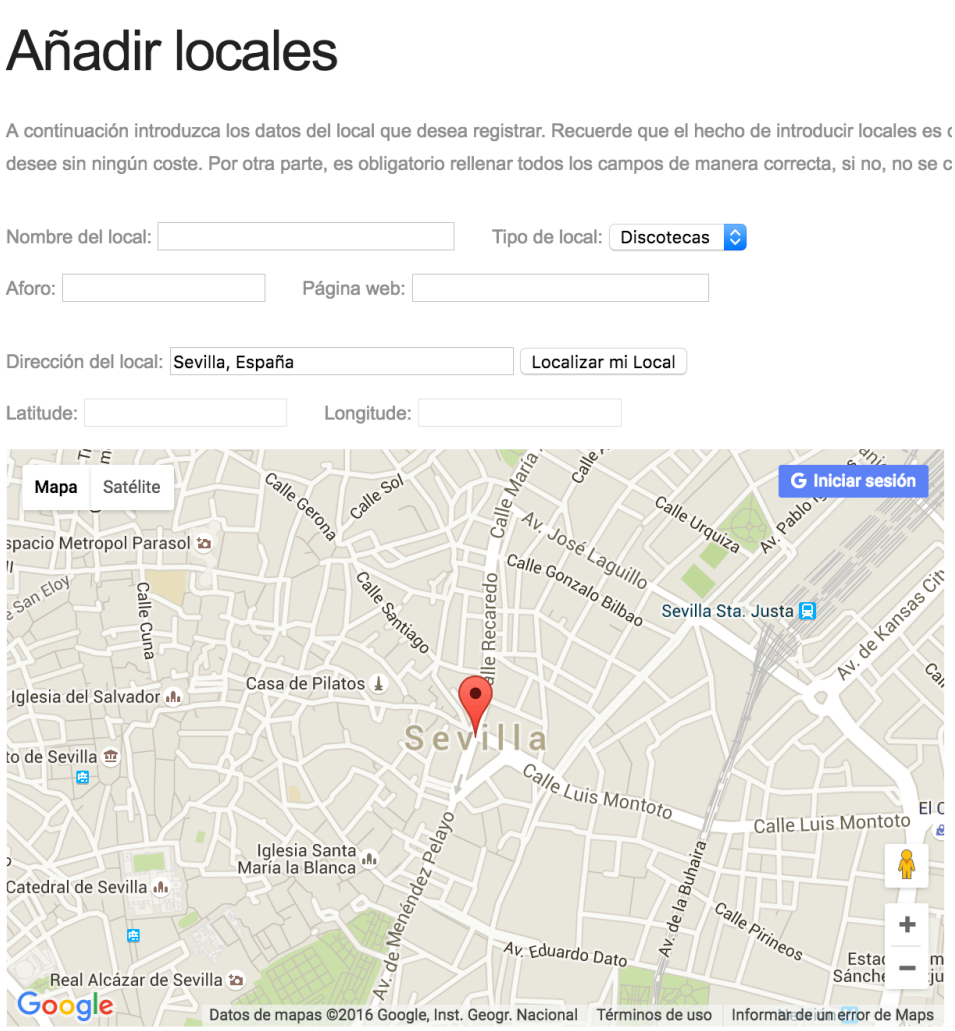


Figura 5-11 Fragmento de la página en la que se añaden locales

5.5.4 Gestión de eventos de un local

5.5.4.1 Mostrar eventos de un local

En la Figura 5-12 se observa la página en la que se muestra una lista en la que aparecen todos los eventos pertenecientes a un local de un determinado usuario, así como un resumen de la información más importante de los mismos. Esto es lo que se muestra por defecto al pulsar la opción "Ver Eventos" de un local concreto.

Hay varias opciones disponibles:

- Modificar (un evento): Nos ofrece la posibilidad de cambiar la información de un evento.
- Eliminar (un evento): Nos ofrece la posibilidad de eliminar un evento.
- Añadir un evento: Nos ofrece la posibilidad añadir un evento.

Figura 5-12 Fragmento de la página en la que se muestran los eventos de un local

5.5.4.2 Modificar eventos de un local

En la Figura 5-13 se muestra la opción disponible en el caso de que seleccionemos "Modificar" en un evento concreto, de los que aparecen en la lista de la Figura 5-12.

La función principal de esta página es actuar como formulario, por lo que su comportamiento es similar el caso que se especificó en el apartado 5.2 de este capítulo.

Hay dos opciones disponibles:

- Confirmar: Envía los datos del evento si el formato de los mismos es correcto y redirige al usuario a la pantalla de la Figura 5-7.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-7.

Figura 5-13 Fragmento de la página en la que se modifican los datos de un evento

5.5.4.3 Añadir eventos a un local

En la Figura 5-14 se muestra la opción disponible en el caso de que seleccionemos "Añadir un evento" en la Figura 5-12.

La función principal de esta página es actuar como formulario, por lo que su comportamiento es similar el caso que se especificó en el apartado 5.2 de este capítulo.

Hay dos opciones disponibles:

- Confirmar: Envía los datos del eventos si el formato de los mismos es correcto y redirige al usuario a la pantalla de la Figura 5-7.
- Cancelar: Redirige al usuario a la pantalla de la Figura 5-7.

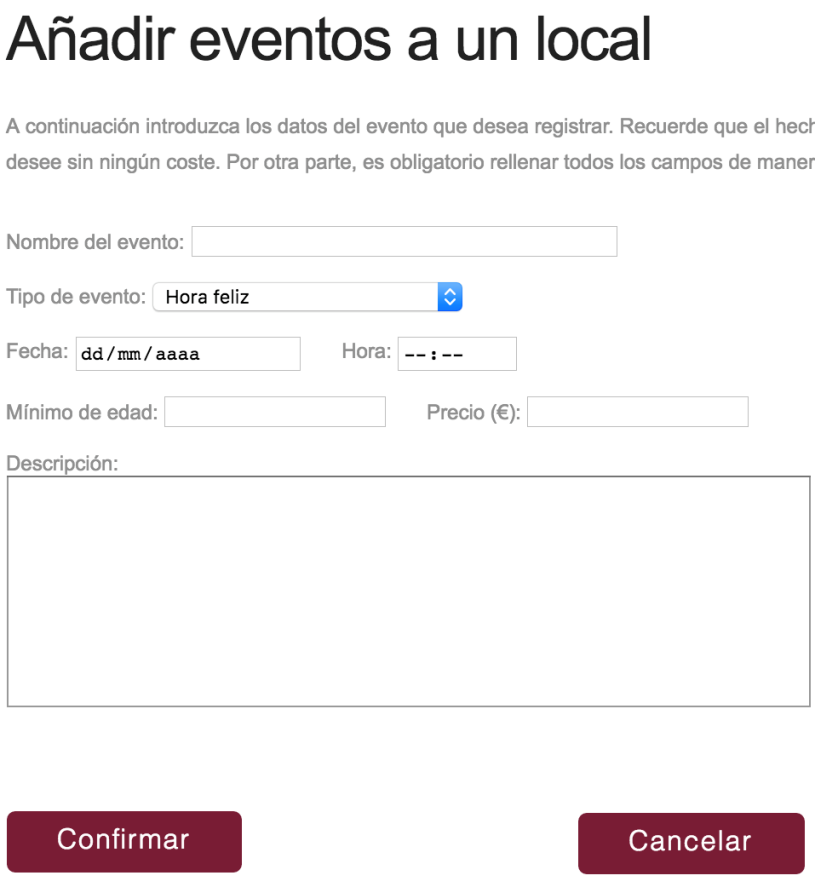


Figura 5-14 Fragmento de la página en la que se añade un evento a un local

5.6 Ayuda al usuario

En este apartado explicaremos las opciones disponibles por parte del usuario en el caso de que seleccione la opción "AYUDA" de la Figura 5-3. En la Figura 5-15 se observa la pantalla que se muestra por defecto en el caso de que se seleccione dicha opción.

Hay dos opciones disponibles:

- Mándanos un correo: Abre una ventana del cliente de correo por defecto del ordenador de usuario, especificando la dirección de correo del Servicio de Atención al Cliente (actualmente no se encuentra en servicio).
- Darme de baja: Redirige al usuario a la pantalla de la Figura 5-17.

Figura 5-15 Fragmento de la página en la que se observa las opciones de la página de Ayuda

5.6.1 Enviar un correo

En la Figura 5-16 se aprecia el comportamiento por defecto. Abre una ventana del cliente de correo que esté configurado por defecto en el ordenador del usuario, especificando la dirección de correo del Servicio de Atención al Cliente (actualmente no se encuentra en servicio).

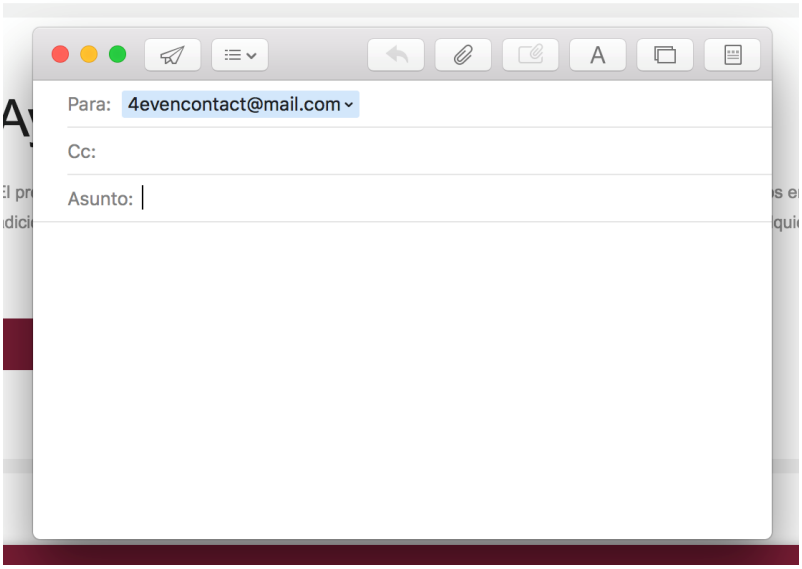


Figura 5-16 Fragmento de la página en la que se observa la ventana del cliente de correo del usuario

5.6.2 Darse de baja del servicio

En este apartado explicaremos las opciones disponibles por parte del usuario en el caso de que seleccione la opción "Darse de baja" de la Figura 5-15. En la Figura 5-17 se observa la pantalla que se muestra por defecto en el caso de que se seleccione dicha opción.

Hay dos opciones disponibles:

- ¡Quiero darme de baja!: Elimina al usuario de la aplicación, así como todos sus locales y eventos. En el caso de que esto se produzca con éxito, aparecerá un mensaje de confirmación como el de la Figura 5-18 y en el caso opuesto, es decir, si se da un error, aparecerá un mensaje con el mismo estilo gráfico, pero informando de dicho error.
- Cancelar: Cancela la operación y redirige al usuario a la pantalla de la Figura 5-5.

Figura 5-17 Fragmento de la página en la que se observa el detalle las opciones al eliminar un perfil

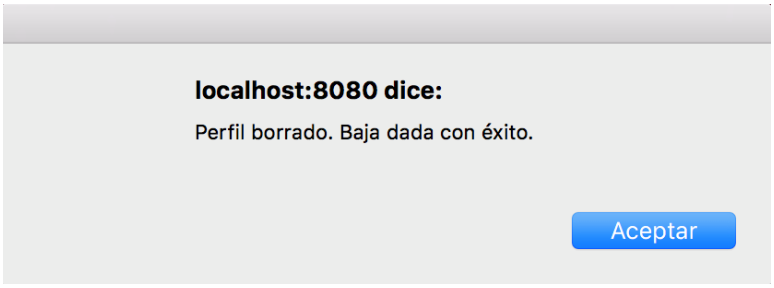


Figura 5-18 Mensaje de alerta en el caso de que el usuario elimine su perfil

5.7 Cierre de sesión de usuario

En el caso de que el usuario seleccione la opción “CERRAR SESIÓN” de la Figura 5-3, aparecerá el mensaje de la figura 5-19, en el caso de que el cierre de sesión se haga de manera correcta. En el caso de que no lo sea, se alertará con un error, que aparecerá en una pantalla que tiene el mismo estilo gráfico.

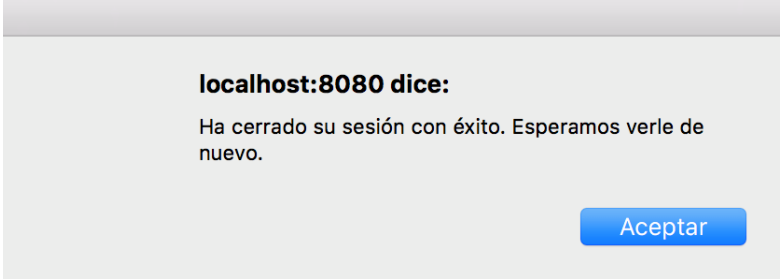


Figura 5-19 Mensaje de alerta en el caso de que el usuario cierre su sesión

6 LÍNEAS DE MEJORA Y CONCLUSIONES

Ningún conocimiento humano puede ir más allá de su experiencia

- John Locke -

En este capítulo nos centraremos en describir las líneas de mejora que en nuestra opinión pueden enriquecer nuestra aplicación, tanto en cuanto a funcionalidad como en tecnologías. Si bien es cierto, tal y como se adivina en la cita superior de este párrafo, que no se puede conocer algo plenamente hasta que se experimenta y pasa a formar parte de la experiencia de uno mismo, nosotros consideraremos la experiencia de terceras personas y opiniones personales, en este apartado, pero que sólo podrán confirmarse cuando se lleven a cabo. También, añadiremos nuestra opinión, experiencia y en definitiva, todas las conclusiones que sacamos en claro, después del desarrollo software que hemos acometido, sobre las herramientas y tecnologías utilizadas.

6.1 Líneas de mejora

6.1.1 Spring Security

6.1.1.1 Motivo

Es una cuestión que sale a relucir en nuestro proyecto el hecho de que no hemos implementado una gran cantidad de funciones de este módulo de Spring Framework, principalmente por la dificultad que entraña la correcta implementación y adaptación del mismo a cualquier aplicación en sí. Pero no es cuestión de Spring Security, sino que en nuestra opinión, los temas de seguridad son complicados y su implementación en servicios web REST requerirían de otro proyecto de fin de grado para realizarlo.

Otro punto importante a tratar sería adaptar nuestro sistema de autenticación para hacerlo compatible completamente con REST, lo cual ahora mismo no sucede, ya que el servicio web tiene que almacenar un valor relacionado con las credenciales de los usuarios que se almacenan en los mismos dentro de la cookie JSESSIONID.

6.1.2 Sistema de gestión de imágenes para los eventos

6.1.2.1 Motivo

Es importante que los usuarios de la aplicación Android dispongan de imágenes en las que se observen cómo transcurrieron eventos que son iguales que el que se va a tener lugar y se anuncian en la aplicación, ya que es sabido que la mayoría de usuarios suelen darle mayor importancia a las fotos que a al resto de información, las cuales a día de hoy, sobre todo entre los jóvenes, tienen una transcendencia muy grande (Facebook, Instagram o Snapchat son de las páginas web más visitadas por jóvenes y su contenido es casi exclusivamente audiovisual).

En un principio estas imágenes serían subidas por el propietario ya que nuestra plataforma no tiene carácter de red social.

6.1.3 Logging y manejo de excepciones: SLF4J

Simple Logging Facade for Java (SLF4J) proporciona una API de registro Java a través de un patrón de fachada. El servidor de registro subyacente se determina en tiempo de implementación y puede ser java.util.logging , log4j, logback o tinylog.s

Figura 6-1 Logo gráfico de SLF4J

La separación de la API de cliente desde el servidor de registro reduce el acoplamiento entre una aplicación y cualquier marco de registro especial. Esto puede hacer más fácil integrar con código existente o de terceros o entregar código en otros proyectos que ya han hecho una opción de registro de back-end.

SLF4J fue creado por Ceki Gülcü como una alternativa más confiable al Jakarta Commons Logging framework (marco de registro/inicio de sesión común de Yakarta).

6.1.3.1 Motivo

La inclusión de un sistema de logging para detectar posibles fallos en un servicio web, relacionados con cualquier aspecto es un punto muy importante en aplicaciones profesionales, con lo que este punto sería el primero a añadir en cuanto a mejora del servicio web, después de securizar la aplicación web en la mejor medida posible.

6.1.4 Documentación del servicio web REST: Swagger

Swagger es una nueva especificación (y su correspondiente implementación para distintas tecnologías) para documentar servicios REST. Swagger proporciona un cliente web para poder acceder de forma muy cómoda y vistosa a la documentación de nuestros servicios.

Figura 6-2 Logo gráfico de Swagger

Además permite probar el servicio desarrollado de forma extremadamente sencilla. En este tutorial veremos cómo documentar y probar nuestros servicios REST con Swagger y las alternativas que ofrece a WSDL o WADL.

6.1.4.1 Motivo

La inclusión de un sistema de documentación como éste sería de gran ayuda en cuanto al trabajo de varios programadores sobre el mismo servicio web, ya que es más amigable gráficamente que otros sistemas de documentación como Javadoc, en el caso del desarrollo en Java.

6.1.5 Compresión y ofuscación del código JS

6.1.5.1 Motivo

Esta tarea es más necesaria de lo que parece en un principio, ya que afecta a dos aspectos importantes de una aplicación web:

- **Compresión**: Lo realizaríamos con el fin de reducir las líneas, espacios y aquellos caracteres no visibles para el programador con el fin de obtener un código que ocupe menos. Hay herramientas online que ya lo hacen, lo que implica es que el código JavaScript que sirves en la aplicación web es diferente del que mantienes, debido a su difícil lectura. Así, disminuiríamos el ancho de banda utilizado por los usuarios, haciendo a su vez que la interfaz web gráfica se cargase más rápido, dando mayor sensación de fluidez.
- **Ofuscación**: Lo que haríamos sería pasar un algoritmo de codificación sobre los ficheros JavaScript para evitar principalmente ataques y vulnerabilidades de seguridad. El fichero JavaScript quedaría ilegible completamente debido a que a diferencia de la compresión, que sólo elimina caracteres no utilizados, la ofuscación lo que hace es asignar a variables letras y números aleatorios.

6.1.6 RESTFUL e Implementación de HATEOAS

6.1.6.1 Introducción

HATEOAS es la abreviación de Hypermedia as the Engine of Application State (hipermedia como motor del estado de la aplicación). Es una restricción de REST que lo distingue de otras arquitecturas. Spring HATEOAS es un módulo que hace uso de esta funcionalidad.

Figura 6-3 Logo gráfico de Spring HATEOAS

El principio es que un cliente interactúa con una aplicación de red completamente a través de hipermedia proporcionadas dinámicamente por los servidores de aplicaciones. Es como que el cliente REST debe ir navegando por la información y no necesita ningún conocimiento previo acerca de la forma de interactuar con cualquier aplicación o servidor más allá de una comprensión genérica de hipermedia.

En otras palabras cuando el servidor nos devuelva la representación de un recurso parte de la información devuelta serán identificadores únicos en forma de hipervínculos a otros recursos asociados.

6.1.6.2 Motivo

Esta tarea no es tan necesaria de acometer como el resto, ya que si bien es cierto, tal y como afirma Roy Fielding en su tesis, que para que un servicio web sea REST debe cumplir todo sus requisitos, en la industria en general se considera esta característica completamente inútil en cuanto a funcionalidad, de hecho el módulo de Spring dedicado al mismo tema, Spring HATEOAS, aún no ha llegado a su versión 1.0. Con todo esto, lo añadiríamos con el fin de que el servicio fuese RESTFUL, debido a que la funcionalidad que añade al mismo es reducida.

6.2 Conclusiones

Una vez dado el proyecto por terminado, pasamos a detallar las conclusiones que podemos observar, ya que ahora disponemos de cierta distancia para contemplar lo que hemos llevado a cabo y de qué manera.

Hemos desarrollado una aplicación con una clara proyección a cualquier usuario y con cierto potencial, que aplicado en la dirección correcta y en nuestra opinión, puede dar como resultado una aplicación con futuro comercial.

La utilidad y su sencillez de uso, así como de diseño, la hacen ideal para cualquier usuario y para gestionar cualquier tipo de evento, haciendo así disposición de una plataforma en la que se promocionen y gestionen eventos, cosa que a día de hoy, no existe tal y como lo hemos concebido nosotros.

Por otra parte, nos gustaría aportar nuestra valoración personal a todo lo que ha supuesto la acometida de este proyecto.

En primer lugar, resaltar el concepto "realista" que nos ha aportado, en el sentido de que en la vida real no se dan las cosas tal y como los problemas que planteamos y resolvemos en clase, ya que normalmente tiene una solución, la cual está ya estudiada, y es sabido que si la aplicamos, funcionará. En la vida real, por el contrario, surgen problemas como los de este trabajo, cuya solución se suele desconocer desde un principio, así como el motivo por el que se producen dichos problemas y es deber nuestro, como ingenieros aprender a resolverlos.

Por otra parte, esto ha resultado en el aprendizaje de una gran cantidad de conceptos de programación y desarrollo, así como de metodologías de trabajo que antes desconocía, conocimientos que a mi parecer complementan perfectamente aquellos que he adquirido en mi experiencia en la Escuela Técnica Superior de Ingeniería.

Finalmente, destacar que todo lo que sacamos de este proyecto es positivo y esperemos que sea útil como aportación para el conocimiento general de la Escuela Técnica Superior de Ingeniería, así como lo ha sido para nosotros.

ANEXO A - DESPLIEGUE EN HEROKU

A.1 Introducción

Heroku (https://www.heroku.com/) es una plataforma de servicio de computación en la Nube que soporta distintos lenguajes de programación.

Figura A-1 Logo gráfico de Heroku

Heroku es propiedad de Salesforce.com y fue desarrollada en 2007 como una de las primeras plataformas de computación en la nube, con el objetivo de soportar solamente el lenguaje de programación Ruby, pero posteriormente se ha extendido el soporte a Java, Node.js, Scala, Clojure, Python y otros lenguajes. Actualmente, el sistema operativo sobre el que funciona Heroku es la distribución Linux Ubuntu.

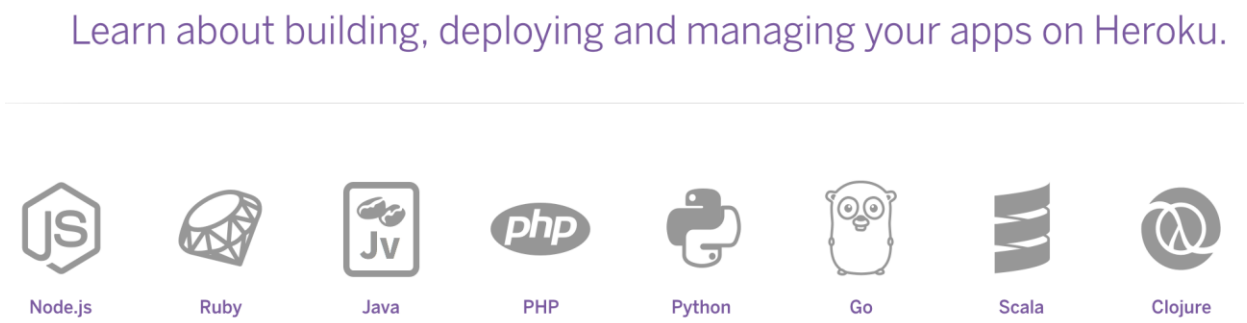


Figura A-2 Lenguages de programación soportados por Heroku

Hay unos elementos especiales en Heroku referentes a la cuestión de la computación: los Dynos. Son piezas fundamentales del modelo de arquitectura de Heroku, ya que son las unidades que proveen la capacidad de cómputo dentro de la plataforma. Están basados en Contenedores Linux.

Cada Dyno está aislado del resto, por lo que los comandos que se ejecutan y los archivos que se almacenan en un Dyno, no afectan a los otros. Además cada Dyno provee el ambiente requerido por las aplicaciones para ser ejecutadas. Se gestionan mediante el Dyno Manager.

A.2 Registro y planes

Heroku dispone de varios planes de suscripción para utilizar su servicio. Sólo uno de ellos es gratuito, por lo que será el que utilicemos para el despliegue de nuestro servicio web.

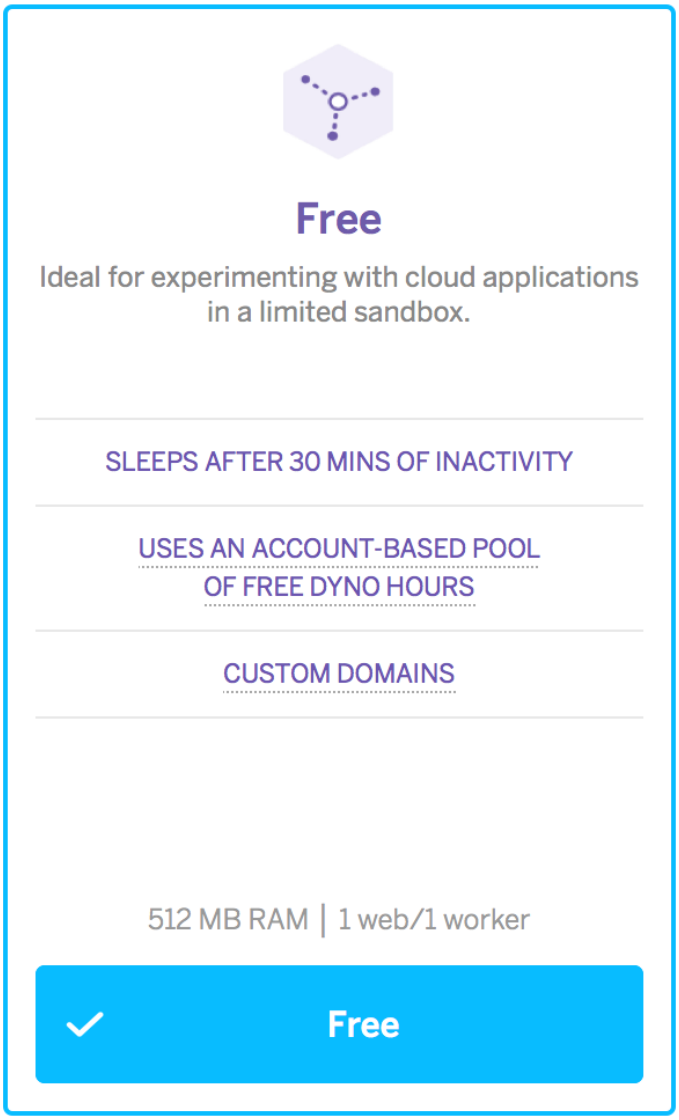


Figura A-3 Plan que hemos elegido para nuestro registro en Heroku

Uno de los puntos positivos que tiene este plan, es que para registrarse y hacer uso del mismo no hace falta darse de alta con una tarjeta de crédito, con lo que nos evita posibles pagos no deseados y lo hacen muy útil para estudiantes.

Darse de alta es sencillo y como hemos dicho previamente, no requiere de ningún tipo de información bancaria durante todo el proceso, incluyendo el despliegue de la aplicación en sí. Esto es importante resaltarlo, ya que hay algunos servicios que mientras el registro es gratuito, una vez intentas desplegar un servicio web, te piden algún tipo de información bancaria con el fin de cobrar alguna cantidad, a pesar de anunciarse como gratuitos desde un principio.

Una vez nos demos de alta, recibiremos un correo electrónico de confirmación y activación a la dirección que

hayamos especificado y una vez activada la cuenta a través de dicho correo, estaremos listos para poder utilizar Heroku.

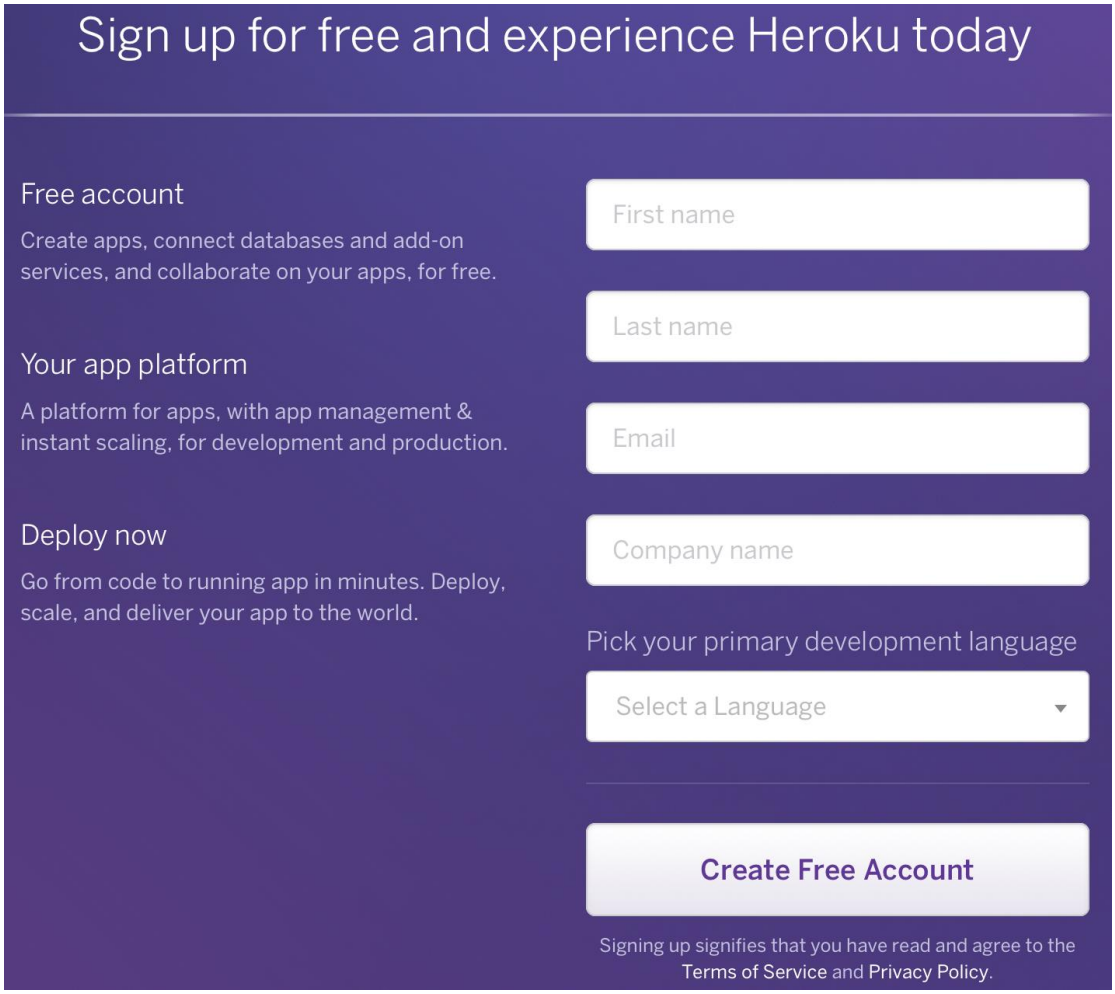


Figura A-4 Formulario de registro de Heroku

A.3 Instalación del cliente local de Heroku en nuestro ordenador

La gestión de nuestras aplicaciones desplegadas en Heroku puede hacerse de dos formas: sin línea de comandos a través de la interfaz gráfica de usuario ofrecida a través de su página web o a través de línea de comandos, haciendo uso de su herramienta Heroku Toolbelt, que deberemos descargar e instalar localmente en nuestro ordenador.

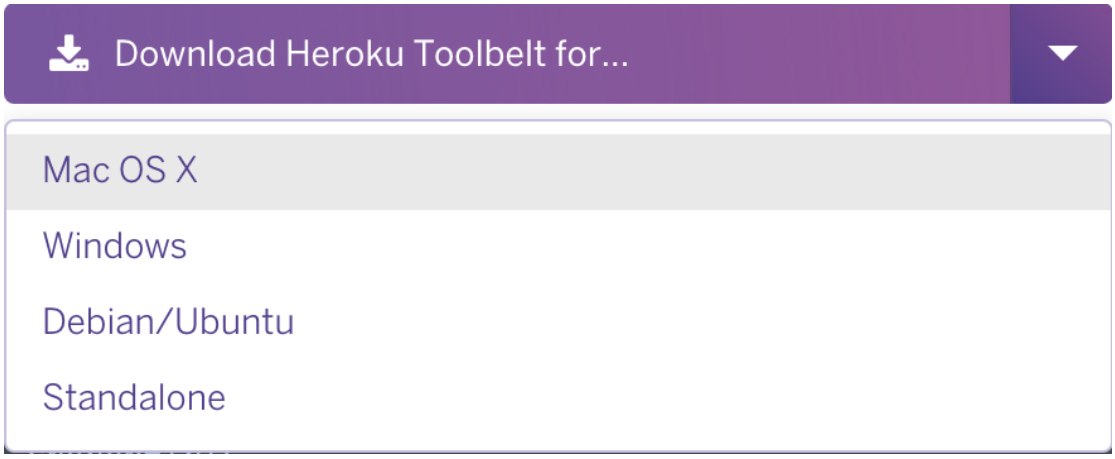


Figura A-5 Sistemas Operativos para los que se encuentra disponible Heroku Toolblet

Una vez descargada e instalada a través de la interfaz gráfica que nos ofrece el ejecutable, ya disponemos de esta herramienta en nuestra máquina local.

Lo siguiente que debemos hacer es escribir en línea de comandos "heroku login" y a continuación, nos pedirá el usuario y contraseña con el que nos registramos en su página web.

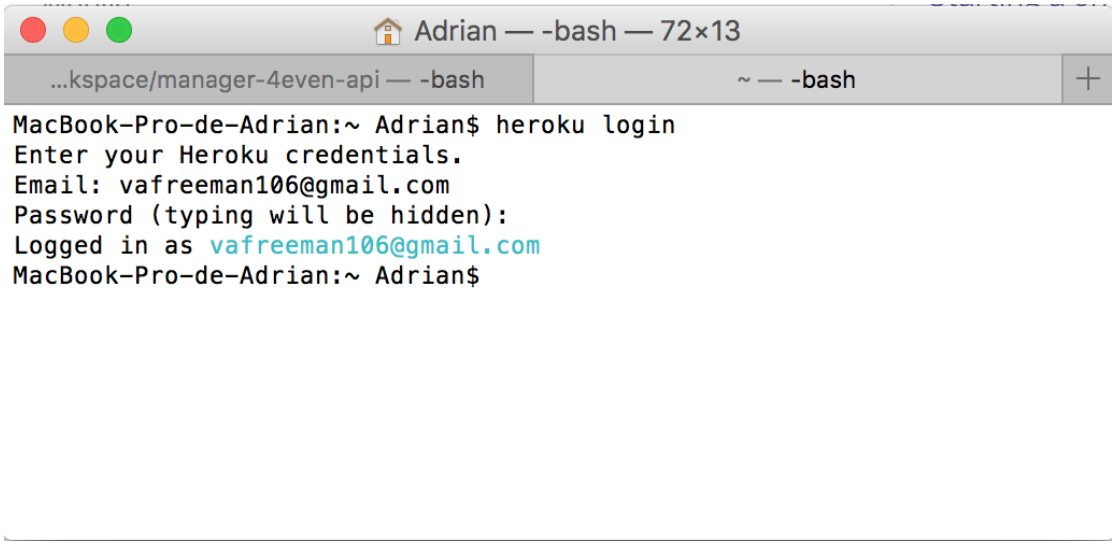


Figura A-6 Login de usuario en el cliente local de Heroku de nuestro ordenador

Una vez iniciada la sesión, podemos proceder a la creación de aplicaciones en Heroku y al despliegue de nuestro servicio web en las mismas.

A.4 Despliegue de la aplicación

El punto más importante por el que hemos elegido Heroku para el despliegue de nuestra aplicación web es porque Heroku soporta de manera nativa el despliegue de aplicaciones basadas en diferentes frameworks y con diferentes características, a diferencia de otras compañías competencia de la que nos ocupa, como por ejemplo OpenShift.

El único prerrequisito que se necesita para proceder al despliegue que no hayamos explicado aún es tener instalado Git, una herramienta de sistema de control de versiones ampliamente utilizada en empresas de desarrollo software. En esta memoria no abordaremos su instalación, ya que en el caso de los sistemas Unix o Linux suele venir preinstalada y en el caso de los sistemas Windows su instalación es trivial.

Debido a la salida en exceso verbosa de la ejecución de los comandos para el despliegue de la aplicación web, no adjuntaremos imágenes, sino directamente los comandos que hemos utilizado, detallando qué operación realiza cada uno.

A.4.1 Pasos para el despliegue de la aplicación

Nº	Descripción del paso a realizar
0	Hemos añadido puntos suspensivos en algunos pasos con el fin de eliminar salida que no es necesaria para entender el proceso de despliegue.
1	Creamos un fichero denominado Procfile y sin extensión, dentro del directorio raíz de nuestro proyecto Spring Boot, con el contenido de abajo, ya que "manager-4even-api-0.1.0.jar" es el fichero .jar de nuestra aplicación Spring Boot. Este fichero Procfile nos ayudará a que el servidor Tomcat embebido escoja el puerto que le obligue Heroku y no el 8080, como hace en los despliegues en nuestra máquina local. `web: java -Dserver.port=$PORT -jar target/manager-4even-api-0.1.0.jar`
2	Nos situamos dentro del directorio raíz de nuestro proyecto Spring. `prompt$: cd ~/Documents/sts-workspace/manager-4even-api/`

3	Mostramos en el directorio en el que estamos.
	`prompt$: pwd`
	`/Users/Adrian/Documents/sts-workspace/manager-4even-api`

4	Mostramos los ficheros que se encuentran en nuestro directorio de trabajo.
	`prompt$: ls –l`
	`Procfile pom.xml src target`

5	Tal y como se comentó, previamente, este es el contenido del fichero Procfile.
	`prompt$: more Procfile`
	`web: java -Dserver.port=$PORT -jar target/manager-4even-api-0.1.0.jar`

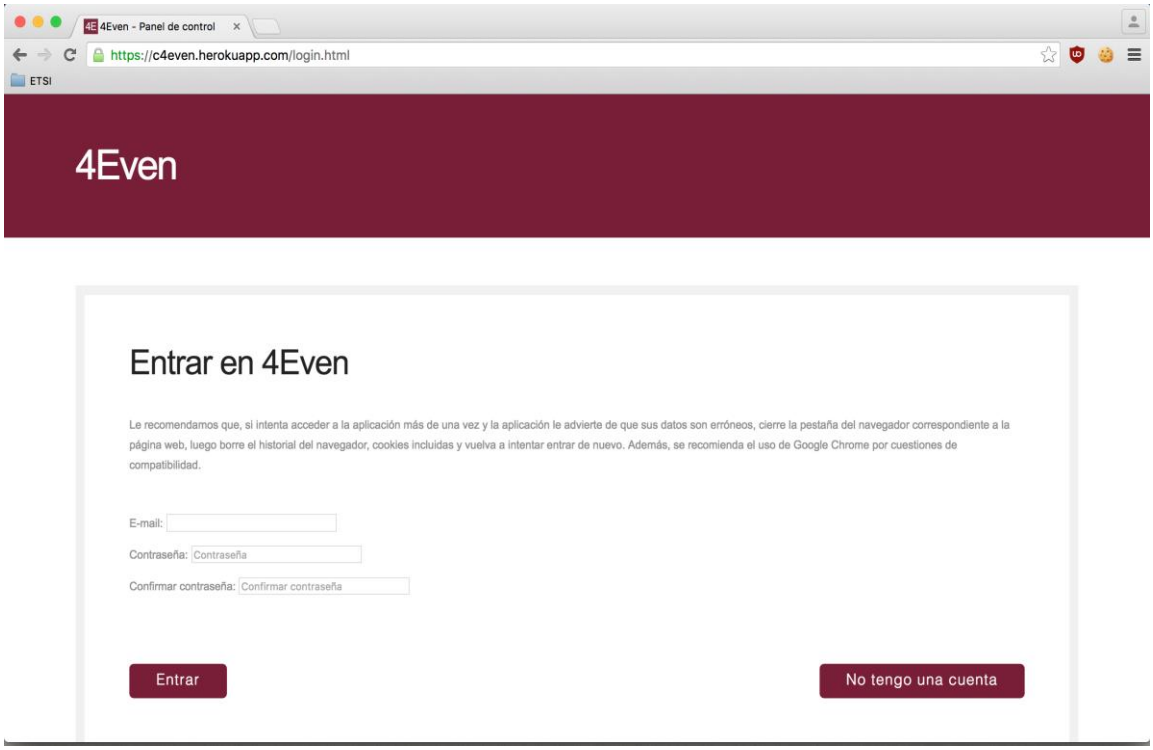
6	Iniciamos un repositorio Git en el directorio raíz de nuestro proyecto.
	`prompt$: git init`
	`Initialized empty Git repository in /Users/Adrian/.Trash/manager-`

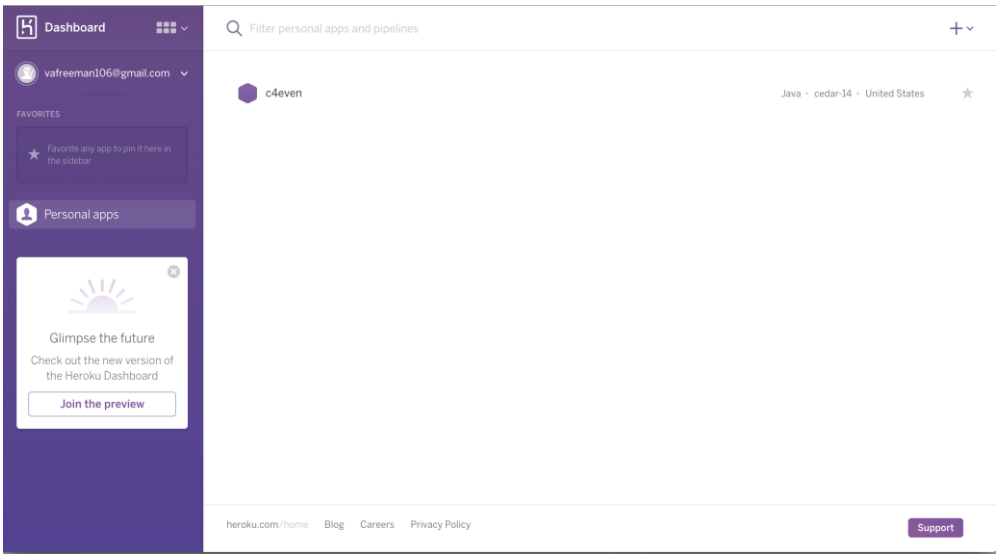
	4even-api/.git/
7	Añadimos todos los ficheros para confirmar los cambios al repositorio. prompt$: git add .
8	Realizamos el primer commit, confirmando la adición de los ficheros añadidos al repositorio previamente. prompt$: git commit -m "Primer commit" [master (root-commit) 3db696b] Primer commit Committer: Adrian <Adrian@MacBook-Pro-de-Adrian.local> Your name and email address were configured automatically based on your username and hostname. Please check that they are accurate. You can suppress this message by setting them explicitly. Run the following command and follow the instructions in your editor to edit your configuration file: git config --global --edit After doing this, you may fix the identity used for this commit with: git commit --amend --reset-author 144 files changed, 7840 insertions(+) create mode 100644 .DS_Store ... create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst
9	Creamos la aplicación Heroku sobre la que desplegaremos el servicio. Por defecto, le asocia un nombre aleatorio, en nuestro caso es nameless-ravine-67668. prompt$: heroku create Creating app... done, □ nameless-ravine-67668 https://nameless-ravine-67668.herokuapp.com/ | https://git.heroku.com/nameless-ravine-67668.git
10	Enviamos los cambios al repositorio remoto de nuestra aplicación que acabamos de crear. Por defecto se envía a la única aplicación activa, en nuestro caso es nameless-ravine-67668 prompt$: git push heroku master Counting objects: 142, done. Delta compression using up to 4 threads. Compressing objects: 100% (127/127), done. Writing objects: 100% (142/142), 18.18 MiB | 454.00 KiB/s, done. Total 142 (delta 45), reused 0 (delta 0) remote: Compressing source files... done. remote: Building source: remote:

	remote: -----> Java app detected remote: -----> Installing OpenJDK 1.8... done remote: -----> Installing Maven 3.3.9... done remote: -----> Executing: mvn -B -DskipTests clean dependency:list install remote: [INFO] Scanning for projects... remote: [INFO] Downloading: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/1.3.3.RELEASE/spring-boot-starter-parent-1.3.3.RELEASE.pom ... remote: Verifying deploy.... done. To https://git.heroku.com/nameless-ravine-67668.git * [new branch] master -> master
11	Con el comando que se adjunta debajo, comprobamos la pantalla de logs que se ejecuta en Spring, al igual que cuando la ejecutamos en nuestra máquina local con el comando java –jar [fichero -jar]. Cualquier fallo de ejecución de la aplicación aparece en esta pantalla, con lo que conviene revisarla al menos durante el despliegue, para comprobar que se ha ejecutado de manera correcta. prompt$: heroku logs --tail 2016-06-01T18:53:12.842441+00:00 app[web.1]: 2016-06-01 18:53:12.837 INFO 3 --- [main] com._4Even.www.Application : Starting Application v0.1.0 on 7912e86d-520b-4430-8e0e-2b9e3a6c7044 with PID 3 (/app/target/manager-4even-api-0.1.0.jar started by u18749 in /app) 2016-06-01T18:53:12.843494+00:00 app[web.1]: 2016-06-01 18:53:12.843 INFO 3 --- [main] com._4Even.www.Application : No active profile set, falling back to default profiles: default ... 2016-06-01T18:53:17.330357+00:00 app[web.1]: 2016-06-01 18:53:17.330 INFO 3 --- [main] e.j.JettyEmbeddedServletContainerFactory : Server initialized with port: 13326 ^C
12	Visualizamos las apps que hemos creado en Heroku. prompt$: heroku apps === My Apps nameless-ravine-67668
13	Ahora que al visualizarla, sabemos el nombre de la app, la renombramos con el nombre c4even, ya que resulta más amigable a la hora operar con ella por línea de comandos. prompt$: heroku apps:rename c4even Renaming nameless-ravine-67668 to c4even... done

	`https://c4even.herokuapp.com/ | https://git.heroku.com/c4even.git`
	`Git remote heroku updated`
	` Don't forget to update git remotes for all other local checkouts of the app.`

14	Comprobamos que la hemos renombrado de manera correcta:
	`prompt$: heroku apps`
	`=== My Apps`
	`c4even`

15	Abrimos la aplicación web con el navegador por defecto de nuestro sistema operativo. `prompt$: heroku open` Y esto es lo que aparece al ejecutar el comando:  **Figura A-7** Servicio web accesible desde Internet

16	Y si comprobamos el dashboard gráfico de la página web de Heroku, observamos que también está nuestra aplicación y podemos gestionarla desde ahí:  **Figura A-8** Gestión de nuestro servicio web desde Heroku

17	Nosotros no lo abordaremos en este anexo, pero pueden añadirse otros añadidos (o add-ons) como bases de datos a nuestra aplicación de manera remota, aunque de manera gratuita la única disponible soporta PostgreSQL, con lo que en nuestro caso habría que migrar las consultas SQL de la aplicación Spring Boot y hacer el script SQL de inicialización de nuestra base de datos MySQL compatible con dicho lenguaje, pero como ya lo hemos comentado, éste no es el tema del anexo que nos ocupa.
	
	Los add-ons disponibles para nuestro proyecto pueden mostrarse mediante el uso del comando:
	
	`prompt$: heroku addons --app c4even`
	
	`Add-on Plan Price`
	`_____ _____ _____`
	`heroku-postgresql (postgresql-objective-30443) hobby-dev free`
	`└── as DATABASE`
	
	`The table above shows add-ons and the attachments to the current app (c4even) or other apps.`

A.4.2 Consideraciones a tener en cuenta

Hay ciertas consideraciones a la hora de operar con Heroku, para aquellas personas que no estén acostumbradas a operar con servicios web en la nube:

- Es posible que al realizar una operación, Heroku tarde entre 1 y 2 minutos en finalizarla, debido al plan gratuito que estamos utilizando y a su carácter de servicio en la nube.
- Es muy importante utilizar la interfaz de gestión por línea de comandos, ya que si gestionamos la aplicación a través de la interfaz web, es altamente posible que surjan conflictos con nuestro repositorio local Git de nuestro proyecto y que dichos conflictos haya que arreglarlos manualmente, modificando los metadatos de Git de este repositorio (que se encuentran en la carpeta .git), lo cual es una tarea que requiere de demasiados conocimientos avanzados.
- A diferencia de otros servicios web en la nube como OpenShift, Heroku no nos obliga a subir el código fuente a plataformas que lo hacen público, como GitHub, con lo que podremos mantener la privacidad del proyecto en el caso de que seamos estudiantes o desarrolladores independientes.

ANEXO B - CÓDIGO DE LA BBDD

4EvenDB.sql

```
-- MySQL Script generated by MySQL Workbench
-- Thu Jun  2 10:19:41 2016
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema 4EvenDB
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `4EvenDB` ;

-- -----------------------------------------------------
-- Schema 4EvenDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `4EvenDB` DEFAULT CHARACTER SET utf8 ;
USE `4EvenDB` ;

-- -----------------------------------------------------
-- Table `4EvenDB`.`OWNERS`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `4EvenDB`.`OWNERS` ;

CREATE TABLE IF NOT EXISTS `4EvenDB`.`OWNERS` (
  `EMAIL` VARCHAR(45) NOT NULL,
  `PASSWORD` VARCHAR(45) NOT NULL,
  `ENABLED` TINYINT(4) NOT NULL,
  PRIMARY KEY (`EMAIL`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `4EvenDB`.`LOCALS`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `4EvenDB`.`LOCALS` ;

CREATE TABLE IF NOT EXISTS `4EvenDB`.`LOCALS` (
  `IDL` INT NOT NULL AUTO_INCREMENT,
  `NAME` VARCHAR(45) NOT NULL,
  `TYPE` VARCHAR(45) NOT NULL,
  `ADDRESS` VARCHAR(100) NOT NULL,
  `LATITUDE` DOUBLE NOT NULL,
  `LONGITUDE` DOUBLE NOT NULL,
  `CAPACITY` INT NOT NULL,
  `DESCRIPTION` VARCHAR(200) NOT NULL,
  `WEBSITE` VARCHAR(45) NOT NULL,
  `OWNERS_EMAIL` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`IDL`, `OWNERS_EMAIL`),
  INDEX `fk_LOCALS_OWNERS_idx` (`OWNERS_EMAIL` ASC),
  CONSTRAINT `fk_LOCALS_OWNERS`
    FOREIGN KEY (`OWNERS_EMAIL`)
    REFERENCES `4EvenDB`.`OWNERS` (`EMAIL`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `4EvenDB`.`EVENTS`
-- -----------------------------------------------------
```

4EvenDB.sql

```
DROP TABLE IF EXISTS `4EvenDB`.`EVENTS` ;

CREATE TABLE IF NOT EXISTS `4EvenDB`.`EVENTS` (
  `IDE` INT NOT NULL AUTO_INCREMENT,
  `NAME` VARCHAR(45) NOT NULL,
  `MIN_AGE` INT NOT NULL,
  `TYPE` VARCHAR(45) NOT NULL,
  `DESCRIPTION` VARCHAR(200) NOT NULL,
  `DATE` DATE NOT NULL,
  `TIME` TIME(6) NOT NULL,
  `PRICE` DOUBLE NOT NULL,
  `LOCALS_IDL` INT NOT NULL,
  `LOCALS_OWNERS_EMAIL` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`IDE`, `LOCALS_IDL`, `LOCALS_OWNERS_EMAIL`),
  INDEX `fk_EVENTS_LOCALS1_idx` (`LOCALS_IDL` ASC, `LOCALS_OWNERS_EMAIL` ASC),
  CONSTRAINT `fk_EVENTS_LOCALS1`
    FOREIGN KEY (`LOCALS_IDL` , `LOCALS_OWNERS_EMAIL`)
    REFERENCES `4EvenDB`.`LOCALS` (`IDL` , `OWNERS_EMAIL`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `4EvenDB`.`OWNERS_ROLES`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `4EvenDB`.`OWNERS_ROLES` ;

CREATE TABLE IF NOT EXISTS `4EvenDB`.`OWNERS_ROLES` (
  `OWNERS_EMAIL` VARCHAR(45) NOT NULL,
  `ROLE` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`OWNERS_EMAIL`),
  CONSTRAINT `fk_OWNERS_ROLES_OWNERS1`
    FOREIGN KEY (`OWNERS_EMAIL`)
    REFERENCES `4EvenDB`.`OWNERS` (`EMAIL`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `4EvenDB`.`OWNERS_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `4EvenDB`.`OWNERS_INFO` ;

CREATE TABLE IF NOT EXISTS `4EvenDB`.`OWNERS_INFO` (
  `OWNERS_EMAIL` VARCHAR(45) NOT NULL,
  `NAME` VARCHAR(45) NOT NULL,
  `PHONENUMBER` VARCHAR(45) NULL,
  `BIRTHDATE` DATE NOT NULL,
  PRIMARY KEY (`OWNERS_EMAIL`),
  CONSTRAINT `fk_OWNERS_INFO_OWNERS1`
    FOREIGN KEY (`OWNERS_EMAIL`)
    REFERENCES `4EvenDB`.`OWNERS` (`EMAIL`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

ANEXO C - CÓDIGO JAVA DEL WS SPRING

Application.java

```java
package com._4Even.www;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.security.config.annotation.web.
configuration.WebSecurityConfigurerAdapter;

/*
 * Clase principal del servicio web, configurada para la inicialización
 * con Spring Boot
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@ComponentScan
@EnableAutoConfiguration
@SpringBootApplication
public class Application {

    @Bean
    public WebSecurityConfigurerAdapter webSecurityConfigurerAdapter() {
    return new SecurityConfiguration();
    }

    public static void main(String args[]) {
    SpringApplication.run(Application.class, args);
    }
}
```

Event.java

```java
package com._4Even.www;

import java.sql.Time;
import java.sql.Date;

/*
 * Clase que representa a un evento en la aplicación.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class Event {
    private String ownerId, eventName, eventType, eventDescription;
    private int localId, eventId, eventMinAge;
    private Date eventDate;
    private Time eventTime;
    private double eventPrice;

    public Event(int eventId, String eventName, int eventMinAge,
        String eventType, String eventDescription, Date eventDate,
        Time eventTime, double eventPrice, int localId, String ownerId) {
    this.ownerId = ownerId;
    this.eventName = eventName;
    this.eventType = eventType;
    this.eventDescription = eventDescription;
    this.localId = localId;
    this.eventId = eventId;
    this.eventMinAge = eventMinAge;
    this.eventDate = eventDate;
    this.eventTime = eventTime;
    this.eventPrice = eventPrice;
    }

    public Event() {
    }

    public String getOwnerId() {
        return ownerId;
    }

    public void setOwnerId(String ownerId) {
        this.ownerId = ownerId;
    }

    public String getEventName() {
        return eventName;
    }

    public void setEventName(String eventName) {
        this.eventName = eventName;
    }

    public String getEventType() {
        return eventType;
    }

    public void setEventType(String eventType) {
        this.eventType = eventType;
```

Event.java

```
    }

    public String getEventDescription() {
        return eventDescription;
    }

    public void setEventDescription(String eventDescription) {
        this.eventDescription = eventDescription;
    }

    public int getLocalId() {
        return localId;
    }

    public void setLocalId(int localId) {
        this.localId = localId;
    }

    public int getEventId() {
        return eventId;
    }

    public void setEventId(int eventId) {
        this.eventId = eventId;
    }

    public int getEventMinAge() {
        return eventMinAge;
    }

    public void setEventMinAge(int eventMinAge) {
        this.eventMinAge = eventMinAge;
    }

    public Date getEventDate() {
        return eventDate;
    }

    public void setEventDate(Date eventDate) {
        this.eventDate = eventDate;
    }

    public Time getEventTime() {
        return eventTime;
    }

    public void setEventTime(Time eventTime) {
        this.eventTime = eventTime;
    }

    public double getEventPrice() {
        return eventPrice;
    }

    public void setEventPrice(double eventPrice) {
        this.eventPrice = eventPrice;
    }

}
```

EventController.java

```java
package com._4Even.www;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ImportResource;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

/*
 * Clase que representa un controlador REST de eventos. Mapea las operaciones
 * sobre recursos REST relacionados con eventos y hace uso del DAO para
 * hacerlas efectivas en la base de datos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@RestController
@ImportResource("classpath:spring/config/beanLocations.xml")
public class EventController {

    // Obtenemos el DAO mediante inyección de dependencias
    @Autowired
    private EventDaoImpl eventDao;

    // Obtiene un evento de la base de datos
    @RequestMapping(value = UriConstants.EVENT, method = RequestMethod.GET)
    public @ResponseBody List<Event> getEvent(
        @PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId,
        @PathVariable("event_id") int eventId) {

    return this.eventDao.getEvent(ownerId, localId, eventId);
    }

    // Obtiene todos los eventos de la base de datos
    @RequestMapping(value = UriConstants.ALL_EVENTS, method = RequestMethod.GET)
    public @ResponseBody List<Event> getAllEvents(
        @PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId) {

    return this.eventDao.getAllEvents(ownerId, localId);
    }

    // Eliminar un evento de la base de datos
    @RequestMapping(value = UriConstants.EVENT, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteEvent(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId,
        @PathVariable("event_id") int eventId) {

    this.eventDao.deleteEvent(ownerId, localId, eventId);
```

EventController.java

```
    }

    // Eliminar todos los eventos de la base de datos
    @RequestMapping(value = UriConstants.ALL_EVENTS, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteAllEvents(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId) {

    this.eventDao.deleteAllEvents(ownerId, localId);
    }

    // Añade un evento en la base de datos
    @RequestMapping(value = UriConstants.ALL_EVENTS, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void addEvent(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId, @RequestBody Event event) {

    this.eventDao.createEvent(event.getEventName(), event.getEventMinAge(),
        event.getEventType(), event.getEventDescription(),
        event.getEventDate(), event.getEventTime(),
        event.getEventPrice(), localId, ownerId);
    }

    // Actualiza un evento en la base de datos
    @RequestMapping(value = UriConstants.EVENT, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updateEvent(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId,
        @PathVariable("event_id") int eventId, @RequestBody Event event) {

    this.eventDao.updateEvent(eventId, event.getEventName(),
        event.getEventMinAge(), event.getEventType(),
        event.getEventDescription(), event.getEventDate(),
        event.getEventTime(), event.getEventPrice(), localId, ownerId);
    }
}
```

EventDao.java

```java
package com._4Even.www;

import java.util.List;
import java.sql.Time;
import java.sql.Date;

/*
 * Clase que representa la interfaz DAO de eventos. Abstrae las operaciones
 * necesarias sobre la base de datos relacionadas con los eventos a petición
 * del controlador REST de eventos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public abstract interface EventDao {
    // No se le pasa el identificador de local para crearlo porque en la base
    // de datos es un campo con autoincremento
    public abstract void createEvent(String eventName, int eventMinAge,
        String eventType, String eventDescription, Date eventDate,
        Time eventTime, double eventPrice, int localId, String ownerId);
    public abstract void updateEvent(int eventId, String eventName,
        int eventMinAge, String eventType, String eventDescription,
        Date eventDate, Time eventTime, double eventPrice, int localId,
        String ownerId);
    public abstract List<Event> getEvent(String ownerId, int localId,
        int eventId);
    public abstract List<Event> getAllEvents(String ownerId, int localId);
    public abstract void deleteEvent(String ownerId, int localId, int eventId);
    public abstract void deleteAllEvents(String ownerId, int localId);
}
```

EventDaoImpl.java

```java
package com._4Even.www;

import java.sql.Date;
import java.sql.Time;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

/*
 * Clase que implementa la interfaz DAO de eventos. Realiza las operaciones
 * necesarias sobre la base de datos relacionadas con los eventos a petición
 * del controlador REST de eventos. No se comentan los métodos debido a que
 * se corresponden con las operaciones del controlador, con lo que ya están
 * descritos previamente
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class EventDaoImpl implements EventDao {

    JdbcTemplate jdbcTemplate;

    public void createEvent(String eventName, int eventMinAge, String eventType,
        String eventDescription, Date eventDate, Time eventTime,
        double eventPrice, int localId, String ownerId) {
    jdbcTemplate.update(SqlConstants.CREATE_EVENT,
        new Object[] { eventName, eventMinAge, eventType,
            eventDescription, eventDate, eventTime, eventPrice,
            localId, ownerId });
    }

    public void updateEvent(int eventId, String eventName, int eventMinAge,
        String eventType, String eventDescription, Date eventDate,
        Time eventTime, double eventPrice, int localId, String ownerId) {
    jdbcTemplate.update(SqlConstants.UPDATE_EVENT,
        new Object[] { eventName, eventMinAge, eventType,
            eventDescription, eventDate, eventTime, eventPrice,
            ownerId, localId, eventId });

    }

    public List<Event> getEvent(String ownerId, int localId, int eventId) {
    return jdbcTemplate.query(SqlConstants.GET_EVENT,
        new Object[] { ownerId, localId, eventId },
        new EventRowMapper());
    }

    public List<Event> getAllEvents(String ownerId, int localId) {
    return jdbcTemplate.query(SqlConstants.GET_ALL_EVENTS,
        new Object[] { ownerId, localId }, new EventRowMapper());
    }

    public void deleteEvent(String ownerId, int localId, int eventId) {
    jdbcTemplate.update(SqlConstants.DELETE_EVENT,
        new Object[] { ownerId, localId, eventId });
    }

    public void deleteAllEvents(String ownerId, int localId) {
```

EventDaoImpl.java

```
    jdbcTemplate.update(SqlConstants.DELETE_ALL_EVENTS,
        new Object[] { ownerId, localId });
    }

    // Inyección del dataSource mediante el constructor
    public void setDataSource(DataSource dataSource) {
    this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public EventDaoImpl() {
    }
}
```

EventRowMapper.java

```java
package com._4Even.www;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/*
 * Clase que representa un mapeador de eventos. Obtiene y crea un evento
 * a partir de los datos obtenidos de la base de datos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class EventRowMapper implements RowMapper<Event> {

    public Event mapRow(ResultSet resultSet, int arg1) throws SQLException {
    // Aquí se hace dentro de cada getX(index) con  el index el número de
    // la columna empezando por la izquierda de la tabla de eventos
    // dentro de la base de datos
    Event event = new Event(resultSet.getInt(1), resultSet.getString(2),
        resultSet.getInt(3), resultSet.getString(4),
        resultSet.getString(5), resultSet.getDate(6),
        resultSet.getTime(7), resultSet.getDouble(8),
        resultSet.getInt(9), resultSet.getString(10));

    return event;
    }
}
```

Local.java

```java
package com._4Even.www;

/*
 * Clase que representa a un local en la aplicación.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class Local {
    private String ownerId, localName, localType, localWebsite,
        localDescription, localAddress;
    private double localLatitude, localLongitude;
    private int localId, localCapacity;

    public Local() {
    }

    public Local(int localId, String localName, String localType,
        String localAddress, double localLatitude, double localLongitude,
        int localCapacity, String localDescription, String localWebsite,
        String ownerId) {
    this.ownerId = ownerId;
    this.localName = localName;
    this.localType = localType;
    this.localWebsite = localWebsite;
    this.localDescription = localDescription;
    this.localAddress = localAddress;
    this.localLatitude = localLatitude;
    this.localLongitude = localLongitude;
    this.localId = localId;
    this.localCapacity = localCapacity;
    }

    public String getOwnerId() {
        return ownerId;
    }

    public void setOwnerId(String ownerId) {
        this.ownerId = ownerId;
    }

    public String getLocalName() {
        return localName;
    }

    public void setLocalName(String localName) {
        this.localName = localName;
    }

    public String getLocalType() {
        return localType;
    }

    public void setLocalType(String localType) {
        this.localType = localType;
    }

    public String getLocalWebsite() {
```

Local.java

```
        return localWebsite;
    }

    public void setLocalWebsite(String localWebsite) {
        this.localWebsite = localWebsite;
    }

    public String getLocalDescription() {
        return localDescription;
    }

    public void setLocalDescription(String localDescription) {
        this.localDescription = localDescription;
    }

    public String getLocalAddress() {
        return localAddress;
    }

    public void setLocalAddress(String localAddress) {
        this.localAddress = localAddress;
    }

    public double getLocalLatitude() {
        return localLatitude;
    }

    public void setLocalLatitude(double localLatitude) {
        this.localLatitude = localLatitude;
    }

    public double getLocalLongitude() {
        return localLongitude;
    }

    public void setLocalLongitude(double localLongitude) {
        this.localLongitude = localLongitude;
    }

    public int getLocalId() {
        return localId;
    }

    public void setLocalId(int localId) {
        this.localId = localId;
    }

    public int getLocalCapacity() {
        return localCapacity;
    }

    public void setLocalCapacity(int localCapacity) {
        this.localCapacity = localCapacity;
    }
}
```

LocalController.java

```java
package com._4Even.www;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ImportResource;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

/*
 * Clase que representa un controlador REST de locales. Mapea las operaciones
 * sobre recursos REST relacionados con locales y hace uso del DAO para
 * hacerlas efectivas en la base de datos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@RestController
@ImportResource("classpath:spring/config/beanLocations.xml")
public class LocalController {

    // Obtenemos el DAO mediante inyección de dependencias
    @Autowired
    private LocalDaoImpl localDao;

    // Obtiene un Local
    @RequestMapping(value = UriConstants.LOCAL, method = RequestMethod.GET)
    public @ResponseBody List<Local> getLocal(
        @PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId) {

    return localDao.getLocal(ownerId, localId);
    }

    // Obtiene todos los Locales
    @RequestMapping(value = UriConstants.ALL_LOCALS, method = RequestMethod.GET)
    public @ResponseBody List<Local> getAllLocals(
        @PathVariable("owner_id") String ownerId) {

    return localDao.getAllLocals(ownerId);
    }

    // Elimina un Local
    @RequestMapping(value = UriConstants.LOCAL, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteLocal(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId) {

    localDao.deleteLocal(ownerId, localId);
    }

    // Elimina todos los Locales
```

LocalController.java

```
    @RequestMapping(value = UriConstants.ALL_LOCALS, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteAllLocals(@PathVariable("owner_id") String ownerId) {

    localDao.deleteAllLocals(ownerId);
    }

    // Añade un Local
    @RequestMapping(value = UriConstants.ALL_LOCALS, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void addLocal(@PathVariable("owner_id") String ownerId,
        @RequestBody Local local) {

    localDao.createLocal(local.getLocalName(), local.getLocalType(),
        local.getLocalAddress(), local.getLocalLatitude(),
        local.getLocalLongitude(), local.getLocalCapacity(),
        local.getLocalDescription(), local.getLocalWebsite(), ownerId);

    }

    // Actualiza un local en la base de datos
    @RequestMapping(value = UriConstants.LOCAL, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updateLocal(@PathVariable("owner_id") String ownerId,
        @PathVariable("local_id") int localId, @RequestBody Local local) {

    localDao.updateLocal(localId, local.getLocalName(),
        local.getLocalType(), local.getLocalAddress(),
        local.getLocalLatitude(), local.getLocalLongitude(),
        local.getLocalCapacity(), local.getLocalDescription(),
        local.getLocalWebsite(), ownerId);

    }
}
```

LocalDao.java

```java
package com._4Even.www;

import java.util.List;

/*
 * Clase que representa la interfaz DAO de locales. Abstrae las operaciones
 * necesarias sobre la base de datos relacionadas con los locales a petición
 * del controlador REST de locales.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public abstract interface LocalDao {
    // No se le pasa el identificador de local para crearlo porque en la base
    // de datos es un campo con autoincremento
    public abstract void createLocal(String localName, String localType,
        String localAddress, double localLatitude, double localLongitude,
        int localCapacity, String localDescription, String localWebsite,
        String ownerId);
    public abstract void updateLocal(int localId, String localName,
        String localType, String localAddress, double localLatitude,
        double localLongitude, int localCapacity, String localDescription,
        String localWebsite, String ownerId);
    public abstract List<Local> getLocal(String ownerId, int localId);
    public abstract List<Local> getAllLocals(String ownerId);
    public abstract void deleteLocal(String ownerId, int localId);
    public abstract void deleteAllLocals(String ownerId);
}
```

LocalDaoImpl.java

```java
package com._4Even.www;

import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

/*
 * Clase que implementa la interfaz DAO de locales. Realiza las operaciones
 * necesarias sobre la base de datos relacionadas con los locales a petición
 * del controlador REST de locales. No se comentan los métodos debido a que
 * se corresponden con las operaciones del controlador, con lo que ya están
 * descritos previamente.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class LocalDaoImpl implements LocalDao {

    JdbcTemplate jdbcTemplate;

    public void createLocal(String localName, String localType,
        String localAddress, double localLatitude, double localLongitude,
        int localCapacity, String localDescription, String localWebsite,
        String ownerId) {
    jdbcTemplate.update(SqlConstants.CREATE_LOCAL,
        new Object[] { localName, localType, localAddress,
            localLatitude, localLongitude, localCapacity,
            localDescription, localWebsite, ownerId });
    }

    public void updateLocal(int localId, String localName, String localType,
        String localAddress, double localLatitude, double localLongitude,
        int localCapacity, String localDescription, String localWebsite,
        String ownerId) {
    jdbcTemplate.update(SqlConstants.UPDATE_LOCAL,
        new Object[] { localName, localType, localAddress,
            localLatitude, localLongitude, localCapacity,
            localDescription, localWebsite, ownerId, localId });
    }

    public List<Local> getLocal(String ownerId, int localId) {
    return jdbcTemplate.query(SqlConstants.GET_LOCAL,
        new Object[] { ownerId, localId }, new LocalRowMapper());
    }

    public List<Local> getAllLocals(String ownerId) {
    return jdbcTemplate.query(SqlConstants.GET_ALL_LOCALS,
        new Object[] { ownerId }, new LocalRowMapper());
    }

    public void deleteLocal(String ownerId, int localId) {
    jdbcTemplate.update(SqlConstants.DELETE_LOCAL_ALL_EVENTS,
        new Object[] { ownerId, localId });
    jdbcTemplate.update(SqlConstants.DELETE_LOCAL,
        new Object[] { ownerId, localId });
    }

    public void deleteAllLocals(String ownerId) {
```

LocalDaoImpl.java

```
    jdbcTemplate.update(SqlConstants.DELETE_ALL_LOCALS_ALL_EVENTS,
        new Object[] { ownerId });
    jdbcTemplate.update(SqlConstants.DELETE_ALL_LOCALS,
        new Object[] { ownerId });
    }

    // Inyección del dataSource mediante el constructor
    public void setDataSource(DataSource dataSource) {
    this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public LocalDaoImpl() {
    }
}
```

LocalRowMapper.java

```java
package com._4Even.www;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/*
 * Clase que representa un mapeador de locales. Obtiene y crea un local
 * a partir de los datos obtenidos de la base de datos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class LocalRowMapper implements RowMapper<Local> {

    public Local mapRow(ResultSet resultSet, int arg1) throws SQLException {
    // Aquí se hace dentro de cada getX(index) con  el index el número de
    // la columna empezando por la izquierda de la tabla de eventos
    // dentro de la base de datos
    Local local = new Local(resultSet.getInt(1), resultSet.getString(2),
        resultSet.getString(3), resultSet.getString(4),
        resultSet.getDouble(5), resultSet.getDouble(6),
        resultSet.getInt(7), resultSet.getString(8),
        resultSet.getString(9), resultSet.getString(10));

    return local;
    }
}
```

Owner.java

```java
package com._4Even.www;

import java.sql.Date;

/*
 * Clase que representa un propietario en la aplicación.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class Owner {
    // El ownerId es el correo electrónico del propietario
    private String ownerId, ownerName, ownerPhoneNumber, ownerPassw;
    private Date ownerBirthDate;

    public Owner() {
    }

    public Owner(String ownerId, String ownerName, String ownerPhoneNumber,
        String ownerPassw, Date ownerBirthDate) {
    this.ownerId = ownerId;
    this.ownerName = ownerName;
    this.ownerPhoneNumber = ownerPhoneNumber;
    this.ownerPassw = ownerPassw;
    this.ownerBirthDate = ownerBirthDate;
    }

    public String getOwnerId() {
    return ownerId;
    }

    public void setOwnerId(String ownerId) {
    this.ownerId = ownerId;
    }

    public String getOwnerName() {
    return ownerName;
    }

    public void setOwnerName(String ownerName) {
    this.ownerName = ownerName;
    }

    public String getOwnerPhoneNumber() {
    return ownerPhoneNumber;
    }

    public void setOwnerPhoneNumber(String ownerPhoneNumber) {
    this.ownerPhoneNumber = ownerPhoneNumber;
    }

    public String getOwnerPassw() {
    return ownerPassw;
    }

    public void setOwnerPassw(String ownerPassw) {
    this.ownerPassw = ownerPassw;
    }
```

Owner.java

```
    public Date getOwnerBirthDate() {
    return ownerBirthDate;
    }

    public void setOwnerBirthDate(Date ownerBirthDate) {
    this.ownerBirthDate = ownerBirthDate;
    }
}
```

OwnerController.java

```java
package com._4Even.www;

import java.util.List;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ImportResource;
import org.springframework.http.HttpStatus;

/*
 * Clase que representa un controlador REST de propeitarios. Mapea las
 * operaciones sobre recursos REST relacionados con eventos y hace uso
 * del DAO para hacerlas efectivas en la base de datos.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@RestController
@ImportResource("classpath:spring/config/beanLocations.xml")
public class OwnerController {

    // Obtenemos el DAO mediante inyección de dependencias
    @Autowired
    private OwnerDaoImpl ownerDao;

    // Obtiene un propietario
    @RequestMapping(value = UriConstants.OWNER, method = RequestMethod.GET)
    public @ResponseBody List<Owner> getOwner(
        @PathVariable("owner_id") String ownerId) {

    return ownerDao.getOwner(ownerId);
    }

    // Obtiene todos los propietarios
    @RequestMapping(value = UriConstants.ALL_OWNERS, method = RequestMethod.GET)
    public @ResponseBody List<Owner> getAllOwners() {

    return ownerDao.getAllOwners();
    }

    // Elimina un propietario
    @RequestMapping(value = UriConstants.OWNER, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteOwner(@PathVariable("owner_id") String ownerId) {

    ownerDao.deleteOwner(ownerId);
    }

    // Elimina todos los propietarios
    @RequestMapping(value = UriConstants.ALL_OWNERS, method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteAllOwners() {
```

OwnerController.java

```
    ownerDao.deleteAllOwners();
    }

    // Añade un propietario
    @RequestMapping(value = UriConstants.OWNER_REGISTER, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void addOwner(@RequestBody Owner owner) {

    ownerDao.createOwner(owner.getOwnerId(), owner.getOwnerName(),
        owner.getOwnerPhoneNumber(), owner.getOwnerBirthDate(),
        owner.getOwnerPassw());
    }

    // Modifica un propietario
    @RequestMapping(value = UriConstants.OWNER, method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updateOwner(@PathVariable("owner_id") String ownerId,
        @RequestBody Owner owner) {

    ownerDao.updateOwner(ownerId, owner.getOwnerName(),
        owner.getOwnerPhoneNumber(), owner.getOwnerBirthDate(),
        owner.getOwnerPassw());
    }
}
```

OwnerDao.java

```java
package com._4Even.www;

import java.sql.Date;
import java.util.List;

/*
 * Clase que representa la interfaz DAO de propietarios. Abstrae las
 * operaciones necesarias sobre la base de datos relacionadas con los
 * propietarios a petición del controlador REST de propietarios.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public abstract interface OwnerDao {
    // Por defecto los propietarios van a crearse con un rol predeterminado
    // ROLE_USER hasta que tengamos disponible la interfaz de gestión de
    // administrador
    // También se habilitan todos por defecto, ya que esto es una función que
    // se encontrará disponible de manera funcional próximamente:
    // ENABLED = 1;
    public abstract void createOwner(String ownerId, String ownerName,
        String ownerPhoneNumber, Date ownerBirthDate, String ownerPassw);
    public abstract void updateOwner(String ownerId, String ownerName,
        String ownerPhoneNumber, Date ownerBirthDate, String ownerPassw);
    public abstract List<Owner> getOwner(String ownerId);
    public abstract List<Owner> getAllOwners();
    public abstract void deleteOwner(String ownerId);
    public abstract void deleteAllOwners();
}
```

OwnerDaoImpl.java

```java
package com._4Even.www;

import java.sql.Date;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

/*
 * Clase que implementa la interfaz DAO de propietarios. Realiza las
 * operaciones necesarias sobre la base de datos relacionadas con los
 * propietarios a petición del controlador REST de propietarios. No se
 * comentan los métodos debido a que se corresponden con las operaciones
 * del controlador, con lo que ya están descritos previamente.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class OwnerDaoImpl implements OwnerDao {

    JdbcTemplate jdbcTemplate;

    public void createOwner(String ownerId, String ownerName,
        String ownerPhoneNumber, Date ownerBirthDate,
        String ownerPassw) {
    // Por defecto, el usuario está habilitado (enabled)
    jdbcTemplate.update(SqlConstants.CREATE_OWNER,
            new Object[] { ownerId, ownerPassw, 1 });
    // No se introduce la contraseña aquí
    jdbcTemplate.update(SqlConstants.CREATE_OWNER_INFO,
        new Object[] { ownerId, ownerName, ownerPhoneNumber,
            ownerBirthDate });
    // Por defecto, el rol es ROLE_USER
    jdbcTemplate.update(SqlConstants.CREATE_OWNER_ROLE,
        new Object[] { ownerId, "ROLE_USER" });
    }

    public void updateOwner(String ownerId, String ownerName,
        String ownerPhoneNumber, Date ownerBirthDate,
        String ownerPassw) {
    jdbcTemplate.update(SqlConstants.UPDATE_OWNER,
            new Object[] { ownerPassw, ownerId });
    jdbcTemplate.update(SqlConstants.UPDATE_OWNER_INFO,
        new Object[] { ownerName, ownerPhoneNumber,
            ownerBirthDate, ownerId });
    // No actualizamos el rol del usuario, ya que es cosa del administrador
    }

    public List<Owner> getOwner(String ownerId) {
    return jdbcTemplate.query(SqlConstants.GET_OWNER_INFO,
        new Object[] { ownerId }, new OwnerRowMapper());
    }

    public List<Owner> getAllOwners() {
    return jdbcTemplate.query(SqlConstants.GET_ALL_OWNERS_INFO,
        new OwnerRowMapper());
    }

    public void deleteOwner(String ownerId) {
```

OwnerDaoImpl.java

```
    jdbcTemplate.update(SqlConstants.DELETE_OWNER_ALL_EVENTS,
        new Object[] { ownerId });
    jdbcTemplate.update(SqlConstants.DELETE_OWNER_ALL_LOCALS,
        new Object[] { ownerId });
    jdbcTemplate.update(SqlConstants.DELETE_OWNER_ROLE,
        new Object[] { ownerId });
    jdbcTemplate.update(SqlConstants.DELETE_OWNER_INFO,
        new Object[] { ownerId });
    jdbcTemplate.update(SqlConstants.DELETE_OWNER,
        new Object[] { ownerId });
    }

    public void deleteAllOwners() {
    jdbcTemplate.update(SqlConstants.DELETE_ALL_OWNERS_ALL_EVENTS);
    jdbcTemplate.update(SqlConstants.DELETE_ALL_OWNERS_ALL_LOCALS);
    jdbcTemplate.update(SqlConstants.DELETE_ALL_OWNERS_ROLE);
    jdbcTemplate.update(SqlConstants.DELETE_ALL_OWNERS_INFO);
    jdbcTemplate.update(SqlConstants.DELETE_ALL_OWNERS);
    }

    // Inyección del dataSource mediante el constructor
    public void setDataSource(DataSource dataSource) {
    this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    public OwnerDaoImpl() {
    }
}
```

OwnerRowMapper.java

```java
package com._4Even.www;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

/*
 * Clase que representa un mapeador de propietarios. Obtiene y crea un
 * propietario a partir de los datos obtenidos de la base de datos.
 * Este RowMapper es para obtener los datos de la tabla OWNERS_INFO
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class OwnerRowMapper implements RowMapper<Owner> {

    public Owner mapRow(ResultSet resultSet, int arg1) throws SQLException {
    // Aquí se hace dentro de cada getX(index) con el index el número de
    // la columna empezando por la izquierda de la tabla de eventos
    // dentro de la base de datos
    //
    // No incluimos la contraseña en el RowMapper
    Owner owner = new Owner(resultSet.getString(1), resultSet.getString(2),
        resultSet.getString(3), "", resultSet.getDate(4));

    return owner;
    }
}
```

RESTAuthenticationEntryPoint.java

```java
package com._4Even.www;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
 * Clase que representa el punto de entrada que se ejecuta
 * en el caso de que un usuario no autenticado intente acceder
 * a nuestro servicio.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@Component
public class RESTAuthenticationEntryPoint implements AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest request,
        HttpServletResponse response,
        AuthenticationException authException)
            throws IOException, ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
            "No tiene autorización para acceder a este contenido");
    }
}
```

RESTAuthenticationFailureHandler.java

```java
package com._4Even.www;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.authentication.
SimpleUrlAuthenticationFailureHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
 * Clase que representa el manejador (handler) que se ejecuta
 * en el caso de que un usuario se autentique de manera incorrecta.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@Component
public class RESTAuthenticationFailureHandler extends
                SimpleUrlAuthenticationFailureHandler {

    @Override
    public void onAuthenticationFailure(HttpServletRequest request,
        HttpServletResponse response,
        AuthenticationException exception)
            throws IOException, ServletException {
        super.onAuthenticationFailure(request, response, exception);
    }
}
```

RESTAuthenticationSuccessHandler.java

```java
package com._4Even.www;

import org.springframework.security.core.Authentication;
import org.springframework.security.web.authentication.
SimpleUrlAuthenticationSuccessHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
 * Clase que representa el manejador (handler) que se ejecuta
 * en el caso de que un usuario se autentique correctamente.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

@Component
public class RESTAuthenticationSuccessHandler extends
                SimpleUrlAuthenticationSuccessHandler {

    @Override
    public void onAuthenticationSuccess(
        HttpServletRequest request,
        HttpServletResponse response,
        Authentication authentication)
            throws IOException, ServletException {
        clearAuthenticationAttributes(request);
    }
}
```

SecurityConfiguration.java

```java
package com._4Even.www;

import com.allanditzel.springframework.security.web.
csrf.CsrfTokenResponseHeaderBindingFilter;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.security.
SecurityProperties;
import org.springframework.core.annotation.Order;
import org.springframework.security.config.annotation.
authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.
web.builders.HttpSecurity;
import org.springframework.security.config.annotation.
web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.web.csrf.CsrfFilter;

/*
 * Clase que implementa toda la seguridad de la aplicación.
 * Incluye aquellos ficheros o URLs de la API que no pueden
 * o pueden visualizarse una vez o no autenticado, además
 * de la protección CSRF.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

// El tipò de autenticación que usamos es FORM-BASED
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Autowired
    private RESTAuthenticationEntryPoint authenticationEntryPoint;
    @Autowired
    private RESTAuthenticationFailureHandler authenticationFailureHandler;
    @Autowired
    private RESTAuthenticationSuccessHandler authenticationSuccessHandler;
    @Autowired
    DataSource dataSource;

    @Override
    protected void configure(AuthenticationManagerBuilder builder)
        throws Exception {
    builder.jdbcAuthentication().dataSource(dataSource)
        .usersByUsernameQuery(SqlConstants.GET_USER_AUTHENTICATION)
        .authoritiesByUsernameQuery(SqlConstants.GET_USER_AUTHORITY);
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
    // Las reglas van de más específicas a menos específicas
    // antMatchers() indica el patrón sobre el que se aplicará la regla
    http.authorizeRequests().antMatchers("/4EvenRegister/**").permitAll();
    http.authorizeRequests()
        .antMatchers("/index.html", "/login.html", "/ownerAdd.html")
        .permitAll();
    http.authorizeRequests().antMatchers("/*.html").authenticated();
```

SecurityConfiguration.java

```
    http.authorizeRequests().antMatchers("/4Even/**").authenticated();
    http.exceptionHandling()
        .authenticationEntryPoint(authenticationEntryPoint);
    http.formLogin().successHandler(authenticationSuccessHandler);
    http.formLogin().failureHandler(authenticationFailureHandler);
    http.logout().logoutSuccessUrl("/");

    // CSRF tokens handling
    http.addFilterAfter(new CsrfTokenResponseHeaderBindingFilter(),
        CsrfFilter.class);
    }
}
```

SqlConstants.java

```java
package com._4Even.www;

/*
 * Clase que almacena las cadenas MySQL de consulta a Base de Datos
 * relacionadas con cada una de las implementaciones DAO de los eventos,
 * locales o propietarios.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class SqlConstants {

    /* AUTHENTICATION */
    public static final String GET_USER_AUTHENTICATION =
        "SELECT EMAIL,PASSWORD,ENABLED FROM OWNERS "
        + "WHERE EMAIL=?";
    public static final String GET_USER_AUTHORITY =
        "SELECT OWNERS_EMAIL,ROLE FROM OWNERS_ROLES "
        + "WHERE OWNERS_EMAIL=?";


    /* OWNERS - OWNERS_INFO - OWNERS_ROLES */
    public static final String CREATE_OWNER =
        "INSERT INTO OWNERS (EMAIL,PASSWORD,ENABLED) VALUES (?,?,?)";
    public static final String CREATE_OWNER_INFO =
        "INSERT INTO OWNERS_INFO (OWNERS_EMAIL,NAME,PHONENUMBER,"
        + "BIRTHDATE) VALUES (?,?,?,?)";
    public static final String CREATE_OWNER_ROLE =
        "INSERT INTO OWNERS_ROLES (OWNERS_EMAIL,ROLE) VALUES (?,?)";

    public static final String UPDATE_OWNER =
        "UPDATE OWNERS SET PASSWORD=? WHERE EMAIL=?";
    public static final String UPDATE_OWNER_INFO =
        "UPDATE OWNERS_INFO SET NAME=?, PHONENUMBER=?, "
        + "BIRTHDATE=? WHERE OWNERS_EMAIL=?";
    // No se actualiza el rol, operación que queda
    // por implementar para futura interfaz de gestión

    public static final String GET_OWNER_INFO =
        "SELECT OWNERS_EMAIL,NAME,PHONENUMBER,BIRTHDATE "
        + "FROM OWNERS_INFO WHERE OWNERS_EMAIL=?";
    public static final String GET_ALL_OWNERS_INFO =
        "SELECT OWNERS_EMAIL,NAME,PHONENUMBER,BIRTHDATE "
        + "FROM OWNERS_INFO";
    public static final String DELETE_OWNER =
        "DELETE FROM OWNERS WHERE EMAIL=?";
    public static final String DELETE_OWNER_INFO =
        "DELETE FROM OWNERS_INFO WHERE OWNERS_EMAIL=?";
    public static final String DELETE_OWNER_ROLE =
        "DELETE FROM OWNERS_ROLES WHERE OWNERS_EMAIL=?";

    public static final String DELETE_OWNER_ALL_LOCALS =
        "DELETE FROM LOCALS WHERE OWNERS_EMAIL=?";
    public static final String DELETE_OWNER_ALL_EVENTS =
        "DELETE FROM EVENTS WHERE LOCALS_OWNERS_EMAIL=?";

    public static final String DELETE_ALL_OWNERS =
        "DELETE FROM OWNERS";
```

SqlConstants.java

```
public static final String DELETE_ALL_OWNERS_INFO =
    "DELETE FROM OWNERS_INFO";
public static final String DELETE_ALL_OWNERS_ROLE =
    "DELETE FROM OWNERS_ROLES";

public static final String DELETE_ALL_OWNERS_ALL_LOCALS =
    "DELETE FROM LOCALS";
public static final String DELETE_ALL_OWNERS_ALL_EVENTS =
    "DELETE FROM EVENTS";


/* LOCALS */
public static final String CREATE_LOCAL =
    "INSERT INTO LOCALS (NAME,TYPE,ADDRESS,LATITUDE,LONGITUDE,"
    + "CAPACITY,DESCRIPTION,WEBSITE,OWNERS_EMAIL) "
    + "VALUES (?,?,?,?,?,?,?,?,?)";

public static final String UPDATE_LOCAL =
    "UPDATE LOCALS SET NAME=?, TYPE=?, ADDRESS=?, LATITUDE=?, "
    + "LONGITUDE=?, CAPACITY=?, DESCRIPTION=?, WEBSITE=? WHERE "
    + "OWNERS_EMAIL=? AND IDL=?";

public static final String GET_LOCAL =
    "SELECT IDL,NAME,TYPE,ADDRESS,LATITUDE,LONGITUDE,CAPACITY,"
    + "DESCRIPTION,WEBSITE,OWNERS_EMAIL FROM LOCALS WHERE "
    + "OWNERS_EMAIL=? AND IDL=?";
public static final String GET_ALL_LOCALS =
    "SELECT IDL,NAME,TYPE,ADDRESS,LATITUDE,LONGITUDE,CAPACITY,"
    + "DESCRIPTION,WEBSITE,OWNERS_EMAIL FROM LOCALS WHERE "
    + "OWNERS_EMAIL=?";

public static final String DELETE_LOCAL =
    "DELETE FROM LOCALS WHERE OWNERS_EMAIL=? AND IDL=?";
public static final String DELETE_LOCAL_ALL_EVENTS =
    "DELETE FROM EVENTS WHERE LOCALS_OWNERS_EMAIL=? "
    + "AND LOCALS_IDL=?";
public static final String DELETE_ALL_LOCALS =
    "DELETE FROM LOCALS WHERE OWNERS_EMAIL=?";
public static final String DELETE_ALL_LOCALS_ALL_EVENTS =
    "DELETE FROM EVENTS WHERE LOCALS_OWNERS_EMAIL=?";


/* EVENTS */
public static final String CREATE_EVENT =
    "INSERT INTO EVENTS (NAME,MIN_AGE,TYPE,DESCRIPTION,DATE,TIME,"
    + "PRICE,LOCALS_IDL,LOCALS_OWNERS_EMAIL) "
    + "VALUES (?,?,?,?,?,?,?,?,?)";

public static final String UPDATE_EVENT =
    "UPDATE EVENTS SET NAME=?, MIN_AGE=?, TYPE=?, DESCRIPTION=?, "
    + "DATE=?, TIME=?, PRICE=? WHERE LOCALS_OWNERS_EMAIL=? AND "
    + "LOCALS_IDL=? AND IDE=?";

public static final String GET_EVENT =
    "SELECT IDE,NAME,MIN_AGE,TYPE,DESCRIPTION,DATE,TIME,"
    + "PRICE,LOCALS_IDL,LOCALS_OWNERS_EMAIL FROM EVENTS WHERE "
    + "LOCALS_OWNERS_EMAIL=? AND LOCALS_IDL=? AND IDE=?";
public static final String GET_ALL_EVENTS =
    "SELECT IDE,NAME,MIN_AGE,TYPE,DESCRIPTION,DATE,TIME,PRICE,"
    + "LOCALS_IDL,LOCALS_OWNERS_EMAIL FROM EVENTS WHERE "
    + "LOCALS_OWNERS_EMAIL=? AND LOCALS_IDL=?";
```

Page 2

SqlConstants.java

```
    public static final String DELETE_EVENT =
        "DELETE FROM EVENTS WHERE LOCALS_OWNERS_EMAIL=? "
        + "AND LOCALS_IDL=? AND IDE=?";
    public static final String DELETE_ALL_EVENTS =
        "DELETE FROM EVENTS WHERE LOCALS_OWNERS_EMAIL=? "
        + "AND LOCALS_IDL=?";

}
```

UriConstants.java

```java
package com._4Even.www;

/*
 * Clase que almacena los URIs accesibles del servicio REST.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */

public class UriConstants {

    /* REGISTER OWNERS */
    // Esta URI no está securizada bajo el sistema de autenticación, con
    // el fin de que usuarios no autenticados puedan darse de alta
    public static final String OWNER_REGISTER =
        "/4EvenRegister//";

    /* OWNERS */
    public static final String OWNER =
        "/4Even/{owner_id:.+}";
    public static final String ALL_OWNERS =
        "/4Even//";

    /* LOCALS */
    public static final String LOCAL =
        "/4Even/{owner_id:.+}/{local_id}";
    public static final String ALL_LOCALS =
        "/4Even/{owner_id:.+}//";

    /* EVENTS */
    public static final String EVENT =
        "/4Even/{owner_id:.+}/{local_id}/{event_id}";
    public static final String ALL_EVENTS =
        "/4Even/{owner_id:.+}/{local_id}//";

}
```

ANEXO D – FICHEROS DE CONFIGURACIÓN WS

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.springframework</groupId>
    <artifactId>manager-4even-api</artifactId>
    <version>1.0</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.3.3.RELEASE</version>
    </parent>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
        </dependency>
        <dependency>
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
          <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-logging</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
        </dependency>
        <dependency>
            <groupId>com.allanditzel</groupId>
            <artifactId>spring-security-csrf-token-filter</artifactId>
            <version>1.1</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
```

pom.xml

```
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

</project>
```

beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

   <bean id="ownerDao" class="com._4Even.www.OwnerDaoImpl" >
        <property name="dataSource" ref="dataSource"></property>
   </bean>
   <bean id="localDao" class="com._4Even.www.LocalDaoImpl" >
        <property name="dataSource" ref="dataSource"></property>
   </bean>
   <bean id="eventDao" class="com._4Even.www.EventDaoImpl" >
        <property name="dataSource" ref="dataSource"></property>
   </bean>
</beans>
```

beanLocations.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <import resource="../beans/beans.xml" />
    <import resource="../database/datasource.xml" />
</beans>
```

datasource.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

    <bean
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="location">
            <value>properties/database.properties</value>
        </property>
    </bean>
    <bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
    </bean>
    <bean id="jdbcTemplate"
    class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
</beans>
```

database.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/4EvenDB
jdbc.username=root
jdbc.password=
```

ANEXO E - CÓDIGO HTML5

eventAdd.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/eventAddController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
```

eventAdd.html

```
<div>
    <!-- Título de la página actual -->
    <h2>Añadir eventos a un local</h2>
    <!-- Breve explicación sobre la página actual -->
    <p>
        A continuación introduzca los datos del evento que
        desea registrar. Recuerde que el hecho de
        introducir eventos es completamente gratuito, por
        lo que puede añadir todos los eventos que
        desee sin ningún coste. Por otra parte, es
        obligatorio rellenar todos los campos de manera
        correcta, si no, no se registrará el evento.
    </p><br/>
    <!-- Formulario en el que introducir
    los datos del evento a añadir -->
    <form id="event_form">
        Nombre del evento:
            <input type="text" name="event_name" size="40"/>
            <br/><br/>
        Tipo de evento:
            <select name="event_type">
                <optgroup label="Alimentación">
                    <option value="Hora feliz">
                    Hora feliz</option>
                    <option value="Desayuno">
                    Desayuno</option>
                    <option value="Almuerzo">
                    Almuerzo</option>
                    <option value="Merienda">
                    Merienda</option>
                    <option value="Cena">
                    Cena</option>
                </optgroup>
                <optgroup label="Aprendizaje de lenguas">
                    <option
                    value="Mesas de intercambio">
                    Mesas de intercambio</option>
                    <option
                    value="Encuentros de intercambio">
                    Encuentros de intercambio</option>
                </optgroup>
                <optgroup label="Encuentros culturales">
                    <option value="Firma de libros">
                    Firma de libros</option>
                    <option value="Turismo por Sevilla">
                    Turismo por Sevilla</option>
                </optgroup>
            </select><br/><br/>
        Fecha:
            <input type="date" name="event_date"/>
             
        Hora:
            <input type="time" name="event_time"/><br/><br/>
        Mínimo de edad:
            <input type="number" name="event_minage"/>
             
        Precio (€):
            <input type="number" name="event_price"/><br/><br/>
        Descripción:<br/>
            <textarea name="event_description" rows="10"
            cols="75" style="resize:none"
            placeholder="Escribe aquí la descripción
```

eventAdd.html

```
                        del evento">
                        </textarea><br/><br/><br/>
                    <!-- Botones de acción del formulario y
                    de cancelación -->
                    <p onclick="submitForm()" class="button-style"
                    style="float:left"><a href="#">Confirmar</a></p>
                    <p class="button-style" style="float:right">
                    <a href="localMain.html">Cancelar</a></p>
                    <br/><br/><br/><br/>
                </form>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

eventMain.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/eventMainController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="narrow-content">
```

eventMain.html

```
                <div>
                    <!-- Título de la página actual -->
                    <h2 class="print-title">Eventos de mi Local </h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        Estos son los locales que tiene registrados
                        en la aplicación, con un resumen de sus datos.
                        En el caso de
                        que quiera añadir nuevos locales o eventos,
                        o modificar los ya existentes, deberá hacer
                        uso de las opciones de debajo de la lista.
                    </p><br/>
                    <!-- Esqueleto de la tabla sobre la que
                    se imprimirán los eventos -->
                    <table class="print-events">
                        <tr>
                            <th>Nombre</th>
                            <th>Tipo</th>
                            <th>Fecha</th>
                            <th>Hora</th>
                            <th>Mín. Edad</th>
                            <th>Precio</th>
                            <th style="text-align:center">
                            Gestionar Evento</th>
                        </tr>
                    </table><br/><br/>
                    <!-- Botones de acción y de cancelación -->
                    <p class="button-style">
                    <a  id="add-event" href="#">
                    Añadir un evento</a></p>
                    <p class="button-style" style="float:right">
                    <a href="localMain.html">
                    Volver</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

eventModify.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/eventModifyController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()">
                <a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
```

eventModify.html

```
<div id="wide-content">
    <div>
        <!-- Título de la página actual -->
        <h2>Modificar eventos de un local</h2>
        <!-- Breve explicación sobre la página actual -->
        <p>
            A continuación introduzca los datos del
            evento que desea registrar. Recuerde que el hecho de
            introducir eventos es completamente gratuito,
            por lo que puede añadir todos los eventos que
            desee sin ningún coste. Por otra parte, es
            obligatorio rellenar todos los campos de manera
            correcta, si no, no se registrará el evento.
        </p><br/>
        <!-- Formulario en el que introducir los datos del
        evento a modificar -->
        <form id="event_form">
            Nombre del evento:
                <input type="text" name="event_name" size="40"/>
                <br/><br/>
            Tipo de evento:
                <select name="event_type">
                    <optgroup label="Alimentación">
                        <option value="Hora feliz">
                        Hora feliz</option>
                        <option value="Desayuno">
                        Desayuno</option>
                        <option value="Almuerzo">
                        Almuerzo</option>
                        <option value="Merienda">
                        Merienda</option>
                        <option value="Cena">
                        Cena</option>
                    </optgroup>
                    <optgroup label="Aprendizaje de lenguas">
                        <option value="Mesas de intercambio">
                        Mesas de intercambio</option>
                        <option value="Encuentros de intercambio">
                        Encuentros de intercambio</option>
                    </optgroup>
                    <optgroup label="Encuentros culturales">
                        <option value="Firma de libros">
                        Firma de libros</option>
                        <option value="Turismo por Sevilla">
                        Turismo por Sevilla</option>
                    </optgroup>
                </select><br/><br/>
            Fecha:
                <input type="date" name="event_date"/>
                 
            Hora:
                <input type="time" name="event_time"/>
                <br/><br/>
            Mínimo de edad:
                <input type="number" name="event_minage"/>
                 
            Precio (€):
                <input type="number" name="event_price"/>
                <br/><br/>
            Descripción<br/>
                <textarea name="event_description" rows="10"
                cols="75" style="resize:none"
```

eventModify.html

```
                        placeholder="Escribe aquí la descripción del
                        evento"></textarea><br/><br/><br/>
                    <!-- Botones de acción del formulario y
                    de cancelación -->
                    <p onclick="submitForm()" class="button-style"
                    style="float:left"><a href="#">Confirmar</a></p>
                    <p class="button-style" style="float:right">
                    <a href="localMain.html">Cancelar</a></p>
                    <br/><br/><br/><br/>
                </form>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

helpMain.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li><a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li class="current_page_item"><a href="helpMain.html"
                accesskey="3">Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
```

helpMain.html

```
                    <h2>Ayuda y contacto</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        El precio de la ayuda en línea y otros servicios
                        relacionados con la atención al usuario,
                        vienen incluidos en el precio de tarifa del
                        servicio, por lo que no tendrá que abonar
                        ningún importe adicional. Le recomendamos que
                        utilice dichos servicios en  el caso de
                        que necesite asistencia en cualquier aspecto
                        relacionado con nuestra aplicación.
                    </p><br/>
                    <!-- Botones de acción del formulario y
                    de cancelación -->
                    <p class="button-style">
                    <a href="mailto:4evencontact@mail.com">
                    Mándanos un correo</a></p>
                    <p class="button-style" style="float:right">
                    <a href="ownerDropOut.html">Darme de baja</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/ownerMainController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li class="current_page_item">
                <a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li><a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
```

index.html

```
                <div>
                    <!-- Título de la página actual -->
                    <h2>Mi Perfil</h2>
                    <!-- Breve descripción de la página web -->
                    <p>
                        A continuación se le muestra su información de
                        perfil, que puede modificar en cualquier
                        momento pulsando en la opción modificar mi perfil.
                        Se recomienda que la información de
                        perfil se encuentre lo más actualizada posible.
                    </p><br/>
                    <!-- Datos de sólo lectura donde se representa la
                    información personal del usuario -->
                    E-mail:
                        <input type="text" name="user_email"
                        size="40" readonly/><br/><br/>
                    Nombre:
                        <input type="text" name="user_name"
                        size="35" readonly/><br/><br/>
                    Fecha de nacimiento:
                        <input type="date" name="user_birthdate"
                        size="10" readonly/><br/><br/>
                    Número de teléfono (opcional):
                        <input type="text" name="user_phonenumber"
                        size="35" readonly/><br/><br/>
                    <!-- Botón de acción sobre datos de usuario-->
                    <p class="button-style"><a href="ownerModify.html">
                    Modificar mi perfil</a></p><br/>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

localAdd.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- API key de acceso a Google Maps -->
<script src="https://maps.googleapis.com/maps/api/js?
key=AIzaSyAWimhGMaKAIvjn_91dymgjY0gHB0lQkew&signed_in=
true&callback=initMap"
async defer></script>

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/maps.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/localAddController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()">
                <a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
```

localAdd.html

```
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
                    <h2>Añadir locales</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        A continuación introduzca los datos
                        del local que desea registrar.
                        Recuerde que el hecho de
                        introducir locales es completamente
                        gratuito, por lo que puede añadir
                        todos los locales que
                        desee sin ningún coste. Por otra parte,
                        es obligatorio rellenar todos los campos
                        de manera correcta, si no, no se
                        completará el registro del local.
                    </p><br/>
                    <!-- Formulario en el que introducir los
                    datos del local a añadir -->
                    <form id="local_form">
                        Nombre del local:
                            <input type="text" name="local_name" size="30"/>
                             
                        Tipo de local:
                            <select name="local_type">
                                <option value="Discotecas">
                                Discotecas</option>
                                <option value="Cines">
                                Cines</option>
                                <option value="Educación">
                                Educación</option>
                                <option value="Conciertos">
                                Conciertos</option>
                                <option value="Especiales">
                                Especiales</option>
                            </select><br/><br/>
                        Aforo:
                            <input type="number" name="local_capacity"/>
                             
                        Página web:
                            <input type="text" name="local_website"
                            size="30"/><br/><br/><br/>
                        Dirección del local:
                            <input id="address" type="text"
                            name="local_address" value="Sevilla, España"
                            size="35"/>
                            <input id="submit" type="button"
                            value="Localizar mi Local"/><br/><br/>
                        Latitude:
                            <input type="text" id="latitude"
                            name="local_latitude" readonly/>
                             
                        Longitude:
                            <input type="text" id="longitude"
                            name="local_longitude" readonly/><br/><br/>
```

localAdd.html

```
                    <!-- Esqueleto del mapa donde representaremos
                    el mapa de Sevilla -->
                    <div id="map"></div><br/><br/><br/>
                    Descripción del local:<br/>
                        <textarea rows="10" cols="75"
                        style="resize:none"
                        placeholder="Escribe aquí la
                        descripción del local" name="local_description">
                        </textarea><br/><br/><br/>
                    <!-- Botones de acción del formulario
                    y de cancelación -->
                    <p onclick="submitForm()" class="button-style"
                    style="float:left"><a href="#">Confirmar</a></p>
                    <p class="button-style" style="float:right">
                    <a href="localMain.html">Cancelar</a></p>
                    <br/><br/><br/><br/>
                </form>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>

</body>
</html>
```

localMain.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/localMainController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="narrow-content">
```

localMain.html

```
                <div>
                    <!-- Título de la página actual -->
                    <h2>Mis Locales</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        Estos son los locales que tiene registrados
                        en la aplicación, con un resumen de sus datos.
                        En el caso de
                        que quiera añadir nuevos locales o eventos,
                        o modificar los ya existentes,
                        deberá hacer uso de las opciones
                        de debajo de la lista.
                    </p><br/>
                    <!-- Esqueleto de la tabla sobre
                    la que se imprimirán los locales -->
                    <table class="print-locals">
                        <tr>
                            <th>Nombre</th>
                            <th>Tipo</th>
                            <th>Dirección</th>
                            <th>Aforo</th>
                            <th>Página web</th>
                            <th style="text-align:center">
                            Gestionar Local</th>
                        </tr>
                    </table><br/><br/><br/><br/>
                    <!-- Botones de acción y de cancelación -->
                    <p class="button-style"><a href="localAdd.html">
                    Añadir un local</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

localModify.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- API key de acceso a Google Maps -->
<script src="https://maps.googleapis.com/maps/api/js?
key=AIzaSyAWimhGMaKAIvjn_91dymgjY0gHB0lQkew&signed_in=
true&callback=initMap"
async defer></script>

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/maps.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/localModifyController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li><a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li class="current_page_item">
                <a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()">
                <a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
```

localModify.html

```
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
                    <h2>Modificar locales</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        A continuación introduzca los datos del local
                         que desea registrar. Recuerde que el hecho de
                        introducir locales es completamente gratuito,
                         por lo que puede añadir todos los locales que
                        desee sin ningún coste. Por otra parte,
                        es obligatorio rellenar todos los campos de
                        manera correcta, si no, no se completará el
                        registro del local.
                    </p><br/>
                    <!-- Formulario en el que introducir los
                    datos del local a añadir -->
                    <form id="local_form">
                        Nombre del local:
                            <input type="text" name="local_name" size="30"/>
                             
                        Tipo de local:
                            <select name="local_type">
                                <option value="Discotecas">
                                Discotecas</option>
                                <option value="Cines">
                                Cines</option>
                                <option value="Educación">
                                Educación</option>
                                <option value="Conciertos">
                                Conciertos</option>
                                <option value="Especiales">
                                Especiales</option>
                            </select><br/><br/>
                        Aforo:
                            <input type="number" name="local_capacity"/>
                             
                        Página web:
                            <input type="text" name="local_website"
                            size="30"/><br/><br/><br/>
                        Dirección del local:
                            <input id="address" type="text"
                            name="local_address" value="Sevilla, España"
                            size="35"/>
                            <input id="submit" type="button"
                            value="Localizar mi Local"/><br/><br/>
                        Latitude:
                            <input type="text" id="latitude"
                            name="local_latitude" readonly/>
                             
                        Longitude:
                            <input type="text" id="longitude"
                            name="local_longitude" readonly/><br/><br/>
                        <!-- Esqueleto del mapa donde representaremos
                        el mapa de Sevilla -->
```

localModify.html

```
                        <div id="map"></div><br/><br/><br/>
                        Descripción del local:<br/>
                            <textarea rows="10" cols="75"
                            style="resize:none"
                            placeholder="Escribe aquí la
                            descripción del local"
                            name="local_description">
                            </textarea><br/><br/><br/>
                        <!-- Botones de acción del formulario
                        y de cancelación -->
                        <p onclick="submitForm()" class="button-style"
                        style="float:left"><a href="#">Confirmar</a></p>
                        <p class="button-style" style="float:right">
                        <a href="localMain.html">Cancelar</a></p>
                        <br/><br/><br/><br/>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/ownerLoginController.js"></script>
</head>

<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
                    <h2>Entrar en 4Even</h2><br/>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        Le recomendamos que, si intenta acceder
                        a la aplicación más de una vez y la aplicación
                        le advierte de que sus datos son erróneos,
                        cierre la pestaña del navegador
                        correspondiente a la página web, luego borre
                        el historial del navegador, cookies
                        incluidas y vuelva a intentar entrar de nuevo.
                        Además, se recomienda el uso de Google
                        Chrome por cuestiones de compatibilidad.
                    </p><br/><br/>
```

login.html

```
                    <!-- Formulario en el que introducir los datos
                    del usuario a autenticar -->
                    <form id="login_form">
                        E-mail:
                            <input type="text" name="user_email"
                            size="30"/>
                        <br/><br/>
                        Contraseña:
                            <input type="password"
                            placeholder="Contraseña" name="user_password"
                            id="user_password" size="30"/>
                        <br/><br/>
                        Confirmar contraseña:
                            <input type="password"
                            placeholder="Confirmar contraseña"
                            name="user_password_confirm" size="30"/>
                        <br/><br/><br/><br/>
                        <!-- Botones de acción del formulario y
                        de cancelación -->
                        <p onclick="submitForm()" class="button-style">
                        <a href="#">Entrar</a></p>
                        <p class="button-style" style="float:right">
                        <a href="ownerAdd.html">No tengo una cuenta</a></p>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

ownerAdd.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/ownerAddController.js"></script>
</head>

<!-- Cuerpo del documento -->
<body>
<div id="header-wrapper">
    <div id="header">
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
                    <h2>Crear una cuenta en 4Even</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        A continuación introduzca sus datos personales,
                         que serán utilizados por 4Even España para
                         identificarle
                        como usuario. Registrarse en la aplicación es
                        completamente gratuito. Por otra parte,
                        es obligatorio rellenar todos los campos de
                        manera correcta, si no, no podrá registrarse
                        como usuario.
                    </p><br/>
                    <!-- Formulario en el que introducir los datos
                    del propietario a añadir -->
                    <form id="register_form">
```

ownerAdd.html

```
                        E-mail:
                            <input type="text" name="user_email"
                            size="40"/><br/><br/>
                        Nombre:
                            <input type="text" name="user_name"
                            size="35"/><br/><br/>
                        Fecha de nacimiento:
                            <input type="date" name="user_birthdate"
                            size="10"/><br/><br/>
                        Número de teléfono (opcional):
                            <input type="text" name="user_phonenumber"
                            size="35"/><br/><br/>
                        Contraseña:
                            <input type="password" placeholder="Contraseña"
                            name="user_password" id="user_password" size="30"/>
                             
                        Confirmar contraseña:
                            <input type="password"
                            placeholder="Confirmar contraseña"
                            name="user_password_confirm" size="30"/>
                            <br/><br/><br/>
                        <!-- Botones de acción del formulario
                        y de cancelación -->
                        <p onclick="submitForm()" class="button-style"
                        style="float:left"><a href="#">Confirmar</a></p>
                        <p class="button-style" style="float:right">
                        <a href="login.html">Cancelar</a></p>
                        <br/><br/><br/><br/>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

ownerDropOut.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/ownerDropOutController.js"></script>
</head>

<!-- Cuerpo del documento -->
<body>
<div id="header-wrapper">
    <div id="header">
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
                <div>
                    <!-- Título de la página actual -->
                    <h2>Darse de baja</h2>
                    <!-- Breve explicación sobre la página actual -->
                    <p>
                        Si se da de baja en el servicio, dejará de
                        disfrutar de todas las facilidades que le
                        da 4Even para la
                        gestión de eventos, debido a que se borrarán
                        todos los datos relacionados con su cuenta personal.
                        Podrá darse de alta posteriormente de nuevo,
                        pero deberá registrar de nuevo sus datos y sus
                        locales a gestionar, así como sus eventos.
                    </p><br/>
                    <!-- Botones de acción del formulario
                    y de cancelación -->
                    <p class="button-style" onclick="deleteUserData()">
```

ownerDropOut.html

```
                    <a href="#">¡Quiero darme de baja!</a></p>
                    <p class="button-style" style="float:right">
                    <a href="helpMain.html">Cancelar</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

ownerModify.html

```
<!DOCTYPE html>
<!--
Diseño por Adrián Gil Gago
Todos los derechos reservados.

Versión: 1.0
-->
<html>

<!-- Cabecera del documento -->
<head>
<!-- Metadatos de la página web -->
<meta charset="UTF-8"/>
<meta name="author" content="Adrián Gil Gago"/>
<meta name="description" content="4Even Gestión de eventos"/>

<!-- Título, fuentes tipográficas y otro contenido enlazado -->
<title>4Even - Panel de control</title>
<link href="css/style.css" rel="stylesheet"/>
<link rel="icon" type="image/png" href="/favicon/favicon.png" />

<!-- Ficheros de código Javascript -->
<script src="js/lib/jquery-1.12.3.min.js"></script>
<script src="js/lib/jquery.validate.min.js"></script>
<script src="js/lib/messages_es.js"></script>
<script src="js/lib/utils.js"></script>
<script src="js/lib/jquery.cookie.js"></script>
<script src="js/ownerModifyController.js"></script>
</head>


<!-- Cuerpo del documento -->
<body>
<!-- Cabecera de la página web -->
<div id="header-wrapper">
    <div id="header">
        <!-- Logo superior izquierdo de 4Even -->
        <div id="logo">
            <h1><a href="#">4Even</a></h1>
        </div>
        <!-- Barra de navegación superior -->
        <div id="menu">
            <ul>
                <li class="current_page_item">
                <a href="index.html" accesskey="1">
                Mi Perfil</a></li>
                <li><a href="localMain.html" accesskey="2">
                Mis Locales</a></li>
                <li><a href="helpMain.html" accesskey="3">
                Ayuda</a></li>
                <li onclick="closeSession()"><a href="#" accesskey="5">
                Cerrar sesión</a></li>
            </ul>
        </div>
    </div>
</div>

<!-- Cuerpo de la página web -->
<div id="wrapper">
    <div id="page-wrapper">
        <div id="page">
            <div id="wide-content">
```

ownerModify.html

```
                <div>
                    <!-- Título de la página actual -->
                    <h2>Modificar mi perfil</h2>
                    <!-- Breve descripción de la página web -->
                    <p>
                        A continuación introduzca los datos del evento
                         que desea registrar. Recuerde que el hecho de
                        introducir eventos es completamente gratuito,
                        por lo que puede añadir todos los eventos que
                        desee sin ningún coste. Por otra parte,
                        es obligatorio rellenar todos los campos de manera
                        correcta, si no, no se registrará el evento.
                    </p><br/>
                    <!-- Formulario en el que introducir los datos
                    del propietario a modificar -->
                    <form id="personal_form">
                        E-mail:
                            <input type="text" name="user_email"
                            size="40"/><br/><br/>
                        Nombre:
                            <input type="text" name="user_name"
                            size="35"/><br/><br/>
                        Fecha de nacimiento:
                            <input type="date" name="user_birthdate"
                            size="10"/><br/><br/>
                        Número de teléfono (opcional):
                            <input type="text" name="user_phonenumber"
                            size="35"/><br/><br/>
                        Contraseña:
                            <input type="password" placeholder="Contraseña"
                            name="user_password" id="user_password"/>
                             
                        Confirmar contraseña:
                            <input type="password"
                            placeholder="Confirmar contraseña"
                            name="user_password_confirm"/><br/><br/><br/>
                        <!-- Botones de acción del formulario y
                        de cancelación -->
                        <p onclick="submitForm()" class="button-style"
                        style="float:left"><a href="#">Confirmar</a></p>
                        <p class="button-style" style="float:right">
                        <a href="index.html">Cancelar</a></p><br/><br/><br/>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- Pie del documento -->
<div id="footer" class="container">
    <p>&copy; 4Even España. Todos los derechos reservados.</p>
</div>
</body>
</html>
```

ANEXO F - CÓDIGO CSS3

style.css

```css
/*
 * Hoja de estilos CSS del FrontEnd web.
 *
 * DiseÃ±o por AdriÃ¡n Gil Gago
 * Todos los derechos reservados.
 * VersiÃ³n: 1.0
 *
 */

body {
    margin: 0px 0px 0px 0px;
    padding: 0;
    background: #781E36;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 12px;
    color: #4b4b4b;
}

h1, h2, h3 {
    margin: 0;
    padding: 0;
    font-weight: 200;
    color: #222222;
}

h2 {
    font-size: 1em;
}

p, ol, ul {
    margin-top: 0px;
    padding: 0px;
}

p, ol {
    line-height: 180%;
}

strong {
}

a {
    color: #5C5539;
}

a:hover {
    text-decoration: none;
}

a img {
    border: none;
}

img.border {
}

img.alignleft {
    float: left;
    margin-top: 8px;
    margin-right: 50px;
}
```

style.css

```css
img.alignright {
    float: right;
}

img.aligncenter {
    margin: 0px auto;
}

hr {
    display: none;
}

/** WRAPPER */

#wrapper {
    overflow: hidden;
    background: #FFFFFF;
    box-shadow:
        inset 0px 11px 8px -20px #CCC,
        inset 0px -11px 8px -10px #CCC;
}

.container {
    width: 1200px;
    margin: 0px auto;
}

.clearfix {
    clear: both;
}

/* Header */

#header-wrapper {
    overflow: hidden;
    height: 150px;
}

#header {
    width: 1200px;
    height: 150px;
    margin: 0 auto;
    padding: 0px 0px;
}

/* Logo */

#logo {
    float: left;
    width: 310px;
    height: 150px;
    padding: 0px 0px 0px 40px;
}

#logo h1 {
    padding: 50px 0px 0px 0px;
    letter-spacing: -2px;
    font-size: 3.6em;
}

#logo h1 a {
```

style.css

```
    text-decoration: none;
    color: #FFFFFF;
}


/* Menu */

#menu {
    float: right;
    width: 810px;
    height: 80px;
    padding: 20px 40px 0px 0px;
}

#menu ul {
    float: right;
    margin: 0;
    padding: 40px 0px 0px 0px;
    list-style: none;
    line-height: normal;
}

#menu li {
    float: left;
}

#menu a {
    display: block;
    margin-left: 1px;
    padding: 7px 20px 7px 20px;
    letter-spacing: 1px;
    text-decoration: none;
    text-align: center;
    text-transform: uppercase;
    font-family: 'Oswald', sans-serif;
    font-size: 16px;
    font-weight: 300;
    color: #FFFFFF;
}

#menu a:hover, #menu .current_page_item a {
    text-decoration: none;
    background: #FFFFFF;
    border-radius: 5px;
    color: #781E36
;
}

/* Page */

#page-wrapper {
    overflow: hidden;
}

#page {
    overflow: hidden;
    width: 1120px;
    margin: 0px auto;
    padding: 50px 40px;
    color: #8F8F8F;
}
```

style.css

```
/** CONTENT */

#wide-content {
    padding: 50px;
    border: 10px solid #f1f1f1;
}

#wide-content h2 {
    padding: 0px 0px 20px 0px;
    letter-spacing: -1px;
    font-size: 36px;
    color: #222222;
}

#narrow-content {
    padding: 50px;
    border: 10px solid #f1f1f1;
}

#narrow-content h2 {
    padding: 0px 0px 20px 0px;
    letter-spacing: -1px;
    font-size: 36px;
    color: #222222;
}


/** CONTENT */

#content {
    float: right;
    width: 760px;
    padding: 0px 0px 0px 0px;
}

#content h2 {
    padding: 0px 0px 20px 0px;
    letter-spacing: -1px;
    font-size: 36px;
    color: #222222;
}

#content .subtitle {
    padding: 0px 0px 30px 0px;
    text-transform: uppercase;
    font-family: 'Oswald', sans-serif;
    font-size: 18px;
    color: #81AFC5;
}

/** SIDEBAR */

#sidebar {
    float: left;
    width: 290px;
    padding: 0px 0px 20px 0px;
}

#sidebar h2 {
    padding: 0px 0px 30px 0px;
    letter-spacing: -1px;
    font-size: 2em;
```

style.css

```
}

/* Footer */

#footer {
    margin: 0 auto;
    padding: 50px 0px 50px 0px;
}

#footer p {
    text-align: center;
    font-size: 12px;
    color: #ffffff;
}

#footer a {
    color: #ffffff;
}

/* Three Column Footer Content */


#footer-bg {
    overflow: hidden;
    padding: 40px 0px;
    background:#E5E0DD;
}

#footer-content {
    color: #67C8E3;
}

#footer-content h2 {
    margin: 0px;
    padding: 0px 0px 30px 0px;
    letter-spacing: -1px;
    text-shadow: 1px 1px 0px #FFFFFF;
    text-transform: uppercase;
    font-size: 20px;
    font-weight: 200;
    color: #93776E;
}

#footer-content a {
}

#footer-content a:hover {
    text-decoration: underline;
}


#column1 {
    float: left;
    width: 280px;
    margin-right: 30px;
}

#column1 p {
    line-height: 1;
}

#column2 {
```

style.css

```
    float: left;
    width: 280px;
    margin-right: 30px;
}

#column3 {
    float: left;
    width: 280px;
}

#column4 {
    float: right;
    width: 260px;
}


/* Button Style */

.button-style {
    display: inline-block;
    margin-top: 30px;
    padding: 7px 30px;
    background: #781E36;
    border-radius: 5px;
}

.button-style a {
    letter-spacing: 1px;
    text-decoration: none;
    font-family: 'Oswald', sans-serif;
    font-weight: 300;
    font-size: 16px;
    color: #FFFFFF;
}

.button-style a:hover
{
    text-decoration: none;
    color: #FFF;
}

/* List Style 1 */

ul.style1 {
    margin: 0px;
    padding: 0px;
    list-style: none;
}

ul.style1 li {
    padding: 35px 0px 25px 0px;
    border-top: 1px solid #c8c8c8;
}

ul.style1 a {
    text-decoration: none;
    color: #FFFFFF;
}

ul.style1 a:hover {
    text-decoration: underline;
    color: #6B6B6B;
```

style.css

```
}

ul.style1 .first {
    padding-top: 0px;
    border-top: none;
}

ul.style1 h3
{
    padding-bottom: 20px;
    font-size: 20px;
    font-weight: bold;
}

ul.style1 .button-style a:hover
{
    text-decoration: none !important;
    color: #FFF !important;
}

/* Button Navbar Style */

.button-style1 {
    margin-top: 30px;
}

.button-style1 a {
    padding: 10px 25px;
    background: #781E36
;
    border-radius: 5px;
    text-decoration: none;
    text-shadow: 1px 1px 0px #FCE3BB;
    color: #36332E !important;
}

/* Form Style */

.form-text-wrapper {
    padding-bottom: 4.5px;
    color: #222222;
    font-size: 120%;
}


.form-data-wrapper {
    border: 2px solid #667172;
    padding-left:3px;
    padding-right: 7px;
    padding-top: 3px;
    padding-bottom: 3px;
    font-size: 100%;
}

#login-form-wrapper {
    text-align: center;
}

/* Map Style */

#map {
    height: 350px;
```

style.css

```
    width: 600px;
    padding-top: 10px;
    padding-bottom: 10px;
}

/* Other Buttons Style */

.button-input {
    display: inline-block;
    margin-top: 30px;
    padding: 3px 30px;
    background: #781E36;
    border-radius: 5px;
    line-height: 180%;
    color: #FFFFFF;
  font-size: 12px;
    letter-spacing: 1px;
  text-decoration: none;
  font-family: 'Oswald', sans-serif;
  font-weight: 300;
  font-size: 16px;
    border-color: #FFFFFF;
}

.mod-buttons {
  color: #FFF;
  background-color: #900;
  font-weight: bold;
    margin: 0 auto;
}

.del-buttons {
    color: #FFF;
    background-color: #900;
    font-weight: bold;
    margin: 0 auto;
    float:right;
}

/* Table Style */

table {
    width:100%;
}

table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}

th, td {
    padding: 5px;
    text-align: left;
}

table.print-locals th   {
    background-color: #575C5D;
    color: white;
}

table.print-locals td   {
    color: black;
```

style.css

```
}

table.print-events th   {
    background-color: #575C5D;
    color: white;
}

table.print-events td   {
    color: black;
}
```

ANEXO G - CÓDIGO DE LIBRERÍAS JS

jquery.cookie.js

```
/*
 * jQuery Cookie Plugin
 * https://github.com/carhartl/jquery-cookie
 *
 *
 * Copyright 2011, Klaus Hartl
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * http://www.opensource.org/licenses/mit-license.php
 * http://www.opensource.org/licenses/GPL-2.0
 */

(function($) {
    $.cookie = function(key, value, options) {

        // key and at least value given, set cookie...
        if (arguments.length > 1 && (!/Object/.
                test(Object.prototype.toString.call(value))
                || value === null || value === undefined)) {
            options = $.extend({}, options);

            if (value === null || value === undefined) {
                options.expires = -1;
            }

            if (typeof options.expires === 'number') {
                var days = options.expires, t = options.expires =
                    new Date();
                t.setDate(t.getDate() + days);
            }

            value = String(value);

            return (document.cookie = [
                encodeURIComponent(key), '=',
                options.raw ? value : encodeURIComponent(value),
                options.expires ? '; expires=' +
                        options.expires.toUTCString() : '',
                // use expires attribute, max-age is not supported by IE
                options.path    ? '; path=' + options.path : '',
                options.domain  ? '; domain=' + options.domain : '',
                options.secure  ? '; secure' : ''
            ].join(''));
        }

        // key and possibly options given, get cookie...
        options = value || {};
        var decode = options.raw ? function(s) { return s; } :
            decodeURIComponent;

        var pairs = document.cookie.split('; ');
        for (var i = 0, pair; pair = pairs[i] &&
        pairs[i].split('='); i++) {
            if (decode(pair[0]) === key) return decode(pair[1] || '');
            // IE saves cookies with empty string as "c; ",
            // e.g. without "=" as opposed to EOMB, thus pair[1]
            // may be undefined
        }
        return null;
    };
})(jQuery);
```

maps.js

```
/*
 * Librería de manejo de mapas para la localización de locales
 * en el lado del cliente, con el fin de facilitar dicha tarea
 * al usuario.
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Función que realiza la inicialización del mapa */
function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom : 15,
        center : {
            lat : 37.389092,
            lng : -5.984459
        }
    });

    var geocoder = new google.maps.Geocoder();
    var defaultLatlng = new google.maps.LatLng(37.389092, -5.984459);
    var marker = new google.maps.Marker({
        map : map,
        position : defaultLatlng
    });

    document.getElementById('submit').addEventListener('click',
            function() {
        geocodeAddress(geocoder, map, showResult, marker);
    });

}


/* Función que imprime los datos de latitud y longitud */
function showResult(result) {
    document.getElementById('latitude').value =
        result.geometry.location.lat();
    document.getElementById('longitude').value =
        result.geometry.location.lng();
}


/* Función que geocodifica la dirección que se le pasa como parámetro */
function geocodeAddress(geocoder, resultsMap, callback, marker) {
    var address = document.getElementById('address').value
            + ", Sevilla, España";

    geocoder.geocode({
        'address' : address
    }, function(results, status) {
        if (status === google.maps.GeocoderStatus.OK) {
            callback(results[0]);
            resultsMap.setCenter(results[0].geometry.location);
            marker.setPosition(results[0].geometry.location);
        } else {
            alert('Ha habido un fallo durante la geocodificación: '
```

maps.js

```
                    + status);
            }
        });
    }
```

utils.js

```
/*
 * Librería de manejo utilidades varias de la página web:
 * incluye una función de cierre de sesión y otra que
 * extrae los parámetros de una URL dada.
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Función que cierra la sesión de usuario */
function closeSession() {
    var cookie = JSON.parse($.cookie('4EvenUser'));
    $.ajax({
        data: {},
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        timeout: 1000,
        type: 'POST',
        url: '/logout'

    }).done(function(data, textStatus, jqXHR) {
        alert("Ha cerrado su sesión con éxito. " +
                "Esperamos verle de nuevo.");
        window.location = '/';

    }).fail(function(jqXHR, textStatus, errorThrown) {
        alert("Ha habido un error en el cierre de su sesión.");
        window.location.href = "index.html";
    });
}

/* Función que extrae los parámetros que se pasan por URL entre páginas */
function getUrlParameter(sParam) {
    // Los parámetros (sParam) deben serpararse por caracteres '&'
    var sPageURL = decodeURIComponent(window.location.search.substring(1)),
    sURLVariables = sPageURL
            .split('&'), sParameterName, i;
    // El valor del parámetro debe ir a continuación del mismo con un =
    for (i = 0; i < sURLVariables.length; i++) {
        sParameterName = sURLVariables[i].split('=');

        if (sParameterName[0] === sParam) {
            return sParameterName[1] === undefined ?
                    true : sParameterName[1];
        }
    }
}
```

ANEXO H - CÓDIGO DE CONTROLADORES JS

eventAddController.js

```
/*
 * Controlador de la página eventAdd.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Inicializamos el plugin de validación
    $('#event_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            event_name : {
                required : true,
                minlength : 2
            },
            event_date : {
                required : true
            },
            event_time : {
                required : true
            },
            event_minage : {
                required : true,
                number : true
            },
            event_price : {
                required : true,
                number : true
            },
            event_description : {
                required : true,
                minlength : 20,
                maxlength : 200
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendEventData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#event_form").submit();
}


/* Función de extracción y envío de los datos del formulario */
function sendEventData() {
    // Obtenemos la cookie
```

eventAddController.js

```
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los parámetros de la URL
    var localId = getUrlParameter('localId');

    // Obtenemos los datos del evento del formulario
    var event_name = $('[name="event_name"]').val();
    var event_type = $('[name="event_type"]').val();
    var event_date = $('[name="event_date"]').val();
    // Añadimos el ':00' por compatibilidad de tipos con la BBDD
    var event_time = $('[name="event_time"]').val() + ':00';
    var event_price = $('[name="event_price"]').val();
    var event_minage = $('[name="event_minage"]').val();
    var event_description = $('[name="event_description"]').val();

    // JSON formado con los datos del formulario extraídos
    var event_json = {
        ownerId : cookie.userid,
        eventName : event_name,
        eventType : event_type,
        eventDescription : event_description,
        localId : localId,
        eventId : '',
        eventMinAge : event_minage,
        eventDate : event_date,
        eventTime : event_time,
        eventPrice : event_price
    };

    // Añadimos el evento a la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId + "/",
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "POST",
        data : JSON.stringify(event_json),
        contentType : "application/json",
        // En caso de éxito: informar y redirigir
        }).done(function (data, textStatus, jqXHR) {
            alert("Evento añadido con éxito");
            window.location.href = "eventMain.html?localId=" + localId;
        // Avisamos al usuario de que ha surgido un error
        }).fail(function (jqXHR, textStatus, errorThrown) {
            alert("Se ha producido un error.");
        });
}
```

eventMainController.js

```
/*
 * Controlador de la página eventMain.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    changeAddButtonUrl();
    getAllEventsData();
});


/* Función que modifica la URL de añadir eventos en
   función del local del que sea el evento*/
function changeAddButtonUrl() {
    var localId = getUrlParameter('localId');
    document.getElementById("add-event").href = "eventAdd.html?localId="
            + localId;
    return false;
};


/* Función que obtiene los datos de todos los eventos
   de un local y de un propietario determinado */
function getAllEventsData(localId) {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    var localId = getUrlParameter('localId');

    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId + "/",
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "GET",
        dataType : "json",
        // En caso de éxito, imprimimos un resumen de los eventos
        success : function(data) {
            printAllEventsData(data);
        },
        // En caso de error: lo mostramos
        error : function(data, status, er) {
            alert("status: " + status + " er:" + er);
        }
    });
}

/* Función que imprime un resumen de todos los eventos de un propietario
   y de un local determinado en una tabla */
function printAllEventsData(jsonEventsArray) {
    // Obtenemos el contenedor donde imprimiremos los eventos
    var container = $(".print-events")[0];
    // Iteramos para cada uno de los eventos e imprimimos sus campos
    for (var i = 0; i < jsonEventsArray.length; i++) {
```

eventMainController.js

```
        var obj = jsonEventsArray[i];
        var summedLocalInfo = "<tr>" + "<td>"
                + obj.eventName
                + "</td>"
                + "<td>"
                + obj.eventType
                + "</td>"
                + "<td>"
                + obj.eventDate
                + "</td>"
                + "<td>"
                + obj.eventTime
                + "</td>"
                + "<td>"
                + obj.eventMinAge
                + "</td>"
                + "<td>"
                + obj.eventPrice
                + " € </td>"
                + "<td>"
                + "<a href='eventModify.html?localId="
                + obj.localId
                + "&eventId="
                + obj.eventId
                + "'><input type='button' class='mod-buttons' "
                + "value='MODIFICAR' /></a>"
                + "<a onclick='deleteEventData("
                + obj.localId
                + ", "
                + obj.eventId
                + ")'><input type='button' class='del-buttons' "
                + "value='ELIMINAR' /></a>"
                + "</td>" + "</tr>"
        container.innerHTML += summedLocalInfo;
    }
}


/* Función que elimina los datos de un evento de la base de datos */
function deleteEventData(localId, eventId) {
    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId + "/" + eventId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "DELETE",
        // En caso de éxito: informar y redirigir
        }).done(function (data, textStatus, jqXHR) {
            alert("Evento borrado.");
            window.location.href = "eventMain.html?localId=" + localId;
            // Avisamos al usuario de que ha surgido un error
        }).fail(function (jqXHR, textStatus, errorThrown) {
            alert("Se ha producido un error.");
        });
}
```

eventModifyController.js

```
/*
 * Controlador de la página eventModify.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Obtenemos los datos del evento de la base de datos
    getEventData();
    // Inicializamos el plugin de validación
    $('#event_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            event_name : {
                required : true,
                minlength : 2
            },
            event_date : {
                required : true
            },
            event_time : {
                required : true
            },
            event_minage : {
                required : true,
                number : true
            },
            event_price : {
                required : true,
                number : true
            },
            event_description : {
                required : true,
                minlength : 20,
                maxlength : 200
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendEventData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#event_form").submit();
}


/* Función de extracción y envío de los datos del formulario */
```

eventModifyController.js

```
function sendEventData() {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los parámetros de la URL
    var localId = getUrlParameter('localId');
    var eventId = getUrlParameter('eventId');

    // Obtenemos los datos del evento del formulario
    var event_name = $('[name="event_name"]').val();
    var event_type = $('[name="event_type"]').val();
    var event_date = $('[name="event_date"]').val();
    var event_time = $('[name="event_time"]').val();
    var event_price = $('[name="event_price"]').val();
    var event_minage = $('[name="event_minage"]').val();
    var event_description = $('[name="event_description"]').val();

    // JSON formado con los datos del formulario extraídos
    var event_json = {
        ownerId : cookie.userid,
        eventName : event_name,
        eventType : event_type,
        eventDescription : event_description,
        localId : localId,
        eventId : eventId,
        eventMinAge : event_minage,
        eventDate : event_date,
        eventTime : event_time,
        eventPrice : event_price
    };

    // Actualizamos la información del evento en la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId + "/" + eventId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "POST",
        data : JSON.stringify(event_json),
        contentType : "application/json",
        // En caso de éxito: informar y redirigir
        success : function() {
            alert("Evento modificado con éxito");
            window.location.href = "eventMain.html?localId=" + localId;
        },
        // En caso de error: mostramos el error
        error : function(status, er) {
            alert("status: " + JSON.stringify(status) + " er:" + er);
        }
    });
}

/* Función que obtiene los datos del evento de la base de datos */
function getEventData() {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los parámetros de la URL
    var localId = getUrlParameter('localId');
    var eventId = getUrlParameter('eventId');
```

eventModifyController.js

```
    // Obtenemos la información del evento de la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId + "/" + eventId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "GET",
        dataType : "json",
        // En caso de éxito: informar y redirigir
        }).done(function (data, textStatus, jqXHR) {
            $('[name="event_name"]').val(data[0].eventName);
            $('[name="event_type"]').val(data[0].eventType);
            $('[name="event_date"]').val(data[0].eventDate);
            $('[name="event_time"]').val(data[0].eventTime);
            $('[name="event_price"]').val(data[0].eventPrice);
            $('[name="event_minage"]').val(data[0].eventMinAge);
            $('[name="event_description"]').val(data[0].eventDescription);
        // Avisamos al usuario de que ha surgido un error
        }).fail(function (jqXHR, textStatus, errorThrown) {
            alert("Se ha producido un error.");
        });
}
```

localAddController.js

```
/*
 * Controlador de la página localAdd.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Inicializamos el plugin de validación
    $('#local_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            local_name : {
                required : true,
                minlength : 2
            },
            local_type : {
            },
            local_capacity : {
                required : true,
                number : true
            },
            local_website : {
                required : true
            },
            local_address : {
                required : true,
                minlength : 2
            },
            local_description : {
                required : true,
                minlength : 20,
                maxlength : 200
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendLocalData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#local_form").submit();
}


/* Función de extracción y envío de los datos del formulario */
function sendLocalData() {

    // Obtenemos la cookie
```

localAddController.js

```
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los datos del evento del formulario
    var local_name = $('[name="local_name"]').val();
    var local_type = $('[name="local_type"]').val();
    var local_capacity = $('[name="local_capacity"]').val();
    var local_website = $('[name="local_website"]').val();
    var local_address = $('[name="local_address"]').val();
    var local_description = $('[name="local_description"]').val();
    var local_longitude = $('[name="local_longitude"]').val();
    var local_latitude = $('[name="local_latitude"]').val();

    // JSON formado con los datos del formulario extraídos
    var local_json = {
        ownerId : cookie.userid,
        localName : local_name,
        localType : local_type,
        localWebsite : local_website,
        localDescription : local_description,
        localAddress : local_address,
        localLatitude : local_latitude,
        localLongitude : local_longitude,
        localId : '',
        localCapacity : local_capacity
    };

    // Añadimos el local a la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/",
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "POST",
        data : JSON.stringify(local_json),
        contentType : "application/json",
    // En caso de éxito: informamos y redirigimos
    }).done(function (data, textStatus, jqXHR) {
        alert("Local añadido con éxito");
        window.location.href = "localMain.html";

    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error.");
    });
}
```

localMainController.js

```
/*
 * Controlador de la página localMain.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    getAllLocalsData();
});


/* Función que obtiene los datos de todos
   los locales de un propietario */
function getAllLocalsData() {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    $.ajax({
        url : "/4Even/" + cookie.userid + "/",
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "GET",
        dataType : "json",
    // En caso de éxito: imprimimos un resumen de los locales
    }).done(function (data, textStatus, jqXHR) {
        printAllLocalsData(data);
    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error");
    });
}


/* Función que imprime un resumen de todos los
   locales de un propietario en una tabla */
function printAllLocalsData(jsonLocalsArray) {
    // Obtenemos el contenedor donde imprimiremos los locales
    var container = $(".print-locals")[0];
    // Iteramos para cada uno de los locales e imprimimos sus campos
    for (var i = 0; i < jsonLocalsArray.length; i++) {
        var obj = jsonLocalsArray[i];
        var summedLocalInfo = "<tr>" + "<td>"
                + obj.localName
                + "</td>"
                + "<td>"
                + obj.localType
                + "</td>"
                + "<td>"
                + obj.localAddress
                + "</td>"
                + "<td>"
                + obj.localCapacity
                + "</td>"
                + "<td>"
                + obj.localWebsite
```

localMainController.js

```
                + "</td>"
                + "<td>"
                + "<a href='localModify.html?localId="
                + obj.localId
                + "'><input type='button' class='mod-buttons'"
                + "value='MODIFICAR' /></a>"
                + "<a href='eventMain.html?localId="
                + obj.localId
                + "'><input type='button' class='mod-buttons' "
                + "value='VER EVENTOS' /></a>"
                + "<a onclick='deleteLocalData("
                + obj.localId
                + ")'><input type='button' class='del-buttons' "
                + "value='ELIMINAR' /></a>"
                + "</td>" + "</tr>"
        container.innerHTML += summedLocalInfo;
    }
}


/* Función que elimina los datos del local de la base de datos */
function deleteLocalData(localId) {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "DELETE",
    // En caso de éxito: informamos y redirigimos
    }).done(function (data, textStatus, jqXHR) {
        alert("Local borrado.");
        window.location.href = "localMain.html";
    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error.");
    });
}
```

localModifyController.js

```
/*
 * Controlador de la página localModify.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Obtenemos los datos del local de la base de datos
    getLocalData();

    // Inicializamos el plugin de validación
    $('#local_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            local_name : {
                required : true,
                minlength : 2
            },
            local_capacity : {
                required : true,
                number : true
            },
            local_website : {
                required : true
            },
            local_address : {
                required : true,
                minlength : 2
            },
            local_description : {
                required : true,
                minlength : 20,
                maxlength : 200
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendLocalData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#local_form").submit();
}


/* Función de extracción y envío de los datos del formulario */
function sendLocalData() {
```

localModifyController.js

```
    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los parámetros de la URL
    var localId = getUrlParameter('localId');

    // Obtenemos los datos local evento del formulario
    var local_name = $('[name="local_name"]').val();
    var local_type = $('[name="local_type"]').val();
    var local_capacity = $('[name="local_capacity"]').val();
    var local_website = $('[name="local_website"]').val();
    var local_address = $('[name="local_address"]').val();
    var local_description = $('[name="local_description"]').val();
    var local_longitude = $('[name="local_longitude"]').val();
    var local_latitude = $('[name="local_latitude"]').val();

    // JSON formado con los datos del formulario extraídos
    var local_json = {
        ownerId : cookie.userid,
        localName : local_name,
        localType : local_type,
        localWebsite : local_website,
        localDescription : local_description,
        localAddress : local_address,
        localLatitude : local_latitude,
        localLongitude : local_longitude,
        localId : localId,
        localCapacity : local_capacity
    };

    // Actualizamos la información del local en la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "POST",
        data : JSON.stringify(local_json),
        contentType : "application/json",
    }).done(function (data, textStatus, jqXHR) {
        alert("Modificación realizada con éxito");
        window.location.href = "localMain.html";
    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error");
    });
}


/* Función que obtiene los datos del local de la base de datos */
function getLocalData() {

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los parámetros de la URL
    var localId = getUrlParameter('localId');

    // Obtenemos la información del evento de la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid + "/"
                + localId,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
```

localModifyController.js

```
        type : "GET",
        dataType : "json",
        cache : false,
    }).done(function (data, textStatus, jqXHR) {
        // Imprimimos los datos en el HTML
        $('[name="local_name"]').val(data[0].localName);
        $('[name="local_type"]').val(data[0].localType);
        $('[name="local_capacity"]').val(data[0].localCapacity);
        $('[name="local_website"]').val(data[0].localWebsite);
        $('[name="local_address"]').val(data[0].localAddress);
        $('[name="local_description"]').val(data[0].localDescription);
        $('[name="local_longitude"]').val(data[0].localLongitude);
        $('[name="local_latitude"]').val(data[0].localLatitude);
    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error.");
    });
}
```

ownerAddController.js

```
/*
 * Controlador de la página ownerAdd.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Inicializamos el plugin de validación
    $('#register_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            user_email : {
                required : true,
                email : true
            },
            user_name : {
                required : true,
                minlength : 2
            },
            user_birthdate : {
                required : true
            },
            user_phonenumber : {
                required : false,
                maxlength: 9,
                minlength : 9
            },
            user_password : {
                required : true,
                minlength : 5
            },
            user_password_confirm : {
                required : true,
                equalTo : "#user_password",
                minlength : 5
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendOwnerData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#register_form").submit();
}


/* Función de extracción y envío de los datos del formulario */
```

ownerAddController.js

```
function sendOwnerData() {

    // Obtenemos los datos del propietario del formulario
    var owner_Id = $('[name="user_email"]').val();
    var owner_Name = $('[name="user_name"]').val();
    var owner_BirthDate = $('[name="user_birthdate"]').val();
    var owner_PhoneNumber = $('[name="user_phonenumber"]').val();
    var owner_Passw = $('[name="user_password"]').val();

    // JSON formado con los datos extraídos del formulario
    var owner_json = {
        ownerId : owner_Id,
        ownerName : owner_Name,
        ownerBirthDate : owner_BirthDate,
        ownerPhoneNumber: owner_PhoneNumber,
        ownerPassw : owner_Passw
    };

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Añadimos el evento a la base de datos
    $.ajax({
        url : "/4EvenRegister/",
        type : "POST",
        data : JSON.stringify(owner_json),
        contentType : "application/json",
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        timeout: 1000
    }).done(function(data, textStatus, jqXHR) {
        alert("Se ha dado de alta con éxito. Bienvenido a 4Even España.");
        window.location.href = "login.html";
    }).fail(function(jqXHR, textStatus, errorThrown) {
        alert("Ha habido un problema en el envío de sus datos.\n " +
                "Le recomendamos que lo intente de nuevo.");
        window.location.href = "ownerAdd.html";
    });
}
```

ownerDropOutController.js

```
/*
 * Controlador de la página ownerDropOut.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


function deleteUserData() {

    // Obtenemos los datos de la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Borramos los datos del usuario de la base de datos
    $.ajax({
        url : "/4Even/" + cookie.userid,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "DELETE",
        // En caso de éxito: informamos y redirigimos
        success : function() {
            alert("Perfil borrado. Baja dada con éxito.");
            window.location.href = "login.html";
        },
        // En caso de error: mostramos el error
        error : function(status, er) {
            alert("status: " + JSON.stringify(status) + " er:" + er);
        }
    });
}
```

ownerLoginController.js

```
/*
 * Controlador de la página ownerLogin.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Inicializamos el plugin de validación
    $('#login_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            user_email : {
                required : true,
                email : true
            },
            user_password : {
                required : true,
                minlength : 5
            },
            user_password_confirm : {
                required : true,
                equalTo : "#user_password",
                minlength : 5
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendOwnerData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#login_form").submit();
}


/* A diferencia del resto de funciones de envíar datos de usuario. ésta lo que
   hace es crear una cookie con el usuario autenticado para no hardcodear el
   usuario dentro del código y su obtención sea dinámica por parte del
   programa,aunque pueda autenticarse cualquier usuario debido a que no
   disponemos de sistema de autenticación.
 */
function sendOwnerData() {

    //showMeYourCookies('At loginform submission');

    // Obtenemos los datos del propietario del formulario
    var owner_Id = $('[name="user_email"]').val();
    var owner_Passw = $('[name="user_password"]').val();
```

ownerLoginController.js

```
    // Obtenemos la cookie de usuario
    var cookie = JSON.parse($.cookie('4EvenUser'));
    var data = 'username=' + owner_Id + '&password=' + owner_Passw;

    $.ajax({
        data: data,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        timeout: 1000,
        type: "POST",
        url: "/login"

    }).done(function(data, textStatus, jqXHR) {
        // No se informa de nada:
        // La autenticación del usuario es autodescriptiva
        // Almacenamos en la cookie 4EvenUser la información del usuario
        var cookie = JSON.stringify({method: '', url: '/', csrf:
jqXHR.getResponseHeader('X-CSRF-TOKEN'), userid: owner_Id});
        $.cookie('4EvenUser', cookie);

        // Obtenemos la cookie nueva
        var cookie = JSON.parse($.cookie('4EvenUser'));

        // Redireccionamos
        window.location = cookie.url;

    }).fail(function(jqXHR, textStatus, errorThrown) {
        // Informamos de que ha fallado la autenticación
        alert("La contraseña o el dni son erróneos");
        // Redirigimos si recarga la página para recargar la cookie
        window.location.href = "index.html";
    });
}
```

ownerMainController.js

```
/*
 * Controlador de la página ownerMain.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Debemos hacer dos consultas: una para obtener el token y otra
    // para obtener los datos de usuario
    getToken();
});

/* Función que obtiene e  imprime los datos del propietario */
function getOwnerData() {

    // Obtenemos la cookie de usuario
    var cookie = JSON.parse($.cookie('4EvenUser'));

    $.ajax({
        url : "/4Even/" + cookie.userid,
        type: 'GET',
        dataType : "json",
    }).done(function (data, textStatus, jqXHR) {
        var csrfToken = jqXHR.getResponseHeader('X-CSRF-TOKEN');
        if (csrfToken) {
            var cookie = JSON.parse($.cookie('4EvenUser'));
            cookie.csrf = csrfToken;
            $.cookie('4EvenUser', JSON.stringify(cookie));
        }
        $('[name="user_email"]').val(data[0].ownerId);
        $('[name="user_name"]').val(data[0].ownerName);
        $('[name="user_birthdate"]').val(data[0].ownerBirthDate);
        $('[name="user_phonenumber"]').val(data[0].ownerPhoneNumber);
    }).fail(function (jqXHR, textStatus, errorThrown) {
        if (jqXHR.status === 401) { // HTTP Status 401: Unauthorized
            var cookie = JSON.stringify({method: '', url: '/', csrf:
jqXHR.getResponseHeader('X-CSRF-TOKEN')});
            $.cookie('4EvenUser', cookie);
            window.location = '/login.html';
        } else {
            alert("Ha habido un error en la obtención de datos de usuario.");
        }
    });
}


/* Función que gestiona el Token de primera conexión */
function getToken() {

    $.ajax({
        url : "/4Even/",
        type: 'GET',
    }).done(function (data, textStatus, jqXHR) {
        var csrfToken = jqXHR.getResponseHeader('X-CSRF-TOKEN');
        if (csrfToken) {
```

ownerMainController.js

```
            var cookie = JSON.parse($.cookie('4EvenUser'));
            cookie.csrf = csrfToken;
            $.cookie('4EvenUser', JSON.stringify(cookie));
        }
        // Sólo una vez autenticados obtenemos los datos de usuario
        getOwnerData();
    }).fail(function (jqXHR, textStatus, errorThrown) {
        if (jqXHR.status === 401) { // HTTP Status 401: Unauthorized
            var cookie = JSON.stringify({method: '', url: '/', csrf:
jqXHR.getResponseHeader('X-CSRF-TOKEN')});
            $.cookie('4EvenUser', cookie);
            window.location = '/login.html';
        } else {
            alert("Ha habido un error en la obtención de datos de usuario.");
        }
    });
}
```

ownerModifyController.js

```
/*
 * Controlador de la página ownerModify.html
 *
 *
 * Diseño por Adrián Gil Gago
 * Todos los derechos reservados.
 * Versión: 1.0
 *
 */


/* Funciones a ejecutar en la carga de la página */
$(document).ready(function() {
    // Obtenemos los datos del local de la base de datos
    getOwnerData();
    // Inicializamos el plugin de validación
    $('#personal_form').validate({
        // Establecemos las reglas de validación para
        // cada uno de los campos del formulario
        rules : {
            user_email : {
                required : true,
                email : true
            },
            user_name : {
                required : true,
                minlength : 2
            },
            user_birthdate : {
                required : true
            },
            user_phonenumber : {
                required : false,
                maxlength: 9,
                minlength : 9
            },
            user_password : {
                required : true,
                minlength : 5
            },
            user_password_confirm : {
                required : true,
                equalTo : "#user_password",
                minlength : 5
            }
        },
        // Establecemos la función que se ejecutará en caso
        // de envío del formulario.
        submitHandler : function(form) {
            sendOwnerData();
        }
    });

});


/* Evento que lanza el envío del formulario */
function submitForm() {
    $("#personal_form").submit();
}
```

ownerModifyController.js

```
/* Función de extracción y envío de los datos del formulario */
function sendOwnerData() {

    // Obtenemos los datos del propietario del formulario
    var owner_Id = $('[name="user_email"]').val();
    var owner_Name = $('[name="user_name"]').val();
    var owner_BirthDate = $('[name="user_birthdate"]').val();
    var owner_PhoneNumber = $('[name="user_phonenumber"]').val();
    var owner_Passw = $('[name="user_password"]').val();

    // JSON formado con los datos extraídos del formulario
    var owner_json = {
        ownerId : owner_Id,
        ownerName : owner_Name,
        ownerBirthDate : owner_BirthDate,
        ownerPhoneNumber: owner_PhoneNumber,
        ownerPassw : owner_Passw
    };

    // Obtenemos la cookie
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Añadimos la información del propietario a la BBDD
    $.ajax({
        url : "/4Even/" + cookie.userid,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "POST",
        data : JSON.stringify(owner_json),
        contentType : "application/json",
        timeout: 1000
    }).done(function(data, textStatus, jqXHR) {
        // Reinicializamos el campo userid de la cookie, por si
        // el usuario lo ha modificado
        cookie.userid = owner_Id;
        // Informamos de la operación y redirigimos
        alert("Modificación realizada con éxito");
        window.location.href = "index.html";
    }).fail(function(jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error.");
    });
}


function getOwnerData() {

    // Obtenemos la cookie de usuario
    var cookie = JSON.parse($.cookie('4EvenUser'));

    // Obtenemos los datos del propietario de la BBDD
    $.ajax({
        url : "/4Even/" + cookie.userid,
        headers: {'X-CSRF-TOKEN': cookie.csrf},
        type : "GET",
        dataType : "json",
    // Imprimimos los datos del propietario en el modelo
    // No imprimimos la contraseña
    }).done(function (data, textStatus, jqXHR) {
        $('[name="user_email"]').val(data[0].ownerId);
        $('[name="user_name"]').val(data[0].ownerName);
        $('[name="user_birthdate"]').val(data[0].ownerBirthDate);
        $('[name="user_phonenumber"]').val(data[0].ownerPhoneNumber);
```

ownerModifyController.js

```
    // Avisamos al usuario de que ha surgido un error
    }).fail(function (jqXHR, textStatus, errorThrown) {
        alert("Se ha producido un error.");
    });
}
```